Superpixel Lattices

Alastair Philip Moore

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Engineering Doctorate

Department of Computer Science

University College London

2011

Abstract

Superpixels are small image segments that are used in popular approaches to object detection and recognition problems. The superpixel approach is motivated by the observation that pixels within small image segments can usually be attributed the same label. This allows a superpixel representation to produce discriminative features based on data dependent regions of support. The reduced set of image primitives produced by superpixels can also be exploited to improve the efficiency of subsequent processing steps. However, it is common for the superpixel representation to have a different graph structure from the original pixel representation of the image.

The first part of the thesis argues that a number of desirable properties of the pixel representation should be maintained by superpixels and that this is not possible with existing methods. We propose a new representation, the *superpixel lattice*, and demonstrate its advantages.

The second part of the thesis investigates incorporating *a priori* information into superpixel segmentations. We learn a probabilistic model that describes the spatial density of object boundaries in the image. We demonstrate our approach using road scene data and show that our algorithm successfully exploits the spatial distribution of object boundaries to improve the superpixel segmentation.

The third part of the thesis presents a globally optimal solution to our superpixel lattice problem in either the horizontal or vertical direction. The solution makes use of a Markov Random Field formulation where the label field is guaranteed to be a set of ordered layers. We introduce an iterative algorithm that uses this framework to learn colour distributions across an image in an unsupervised manner.

We conclude that our approach achieves comparable or better performance than competing methods and that it confers several additional advantages.

Declaration

This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text.

.....

Alastair Moore

Acknowledgements

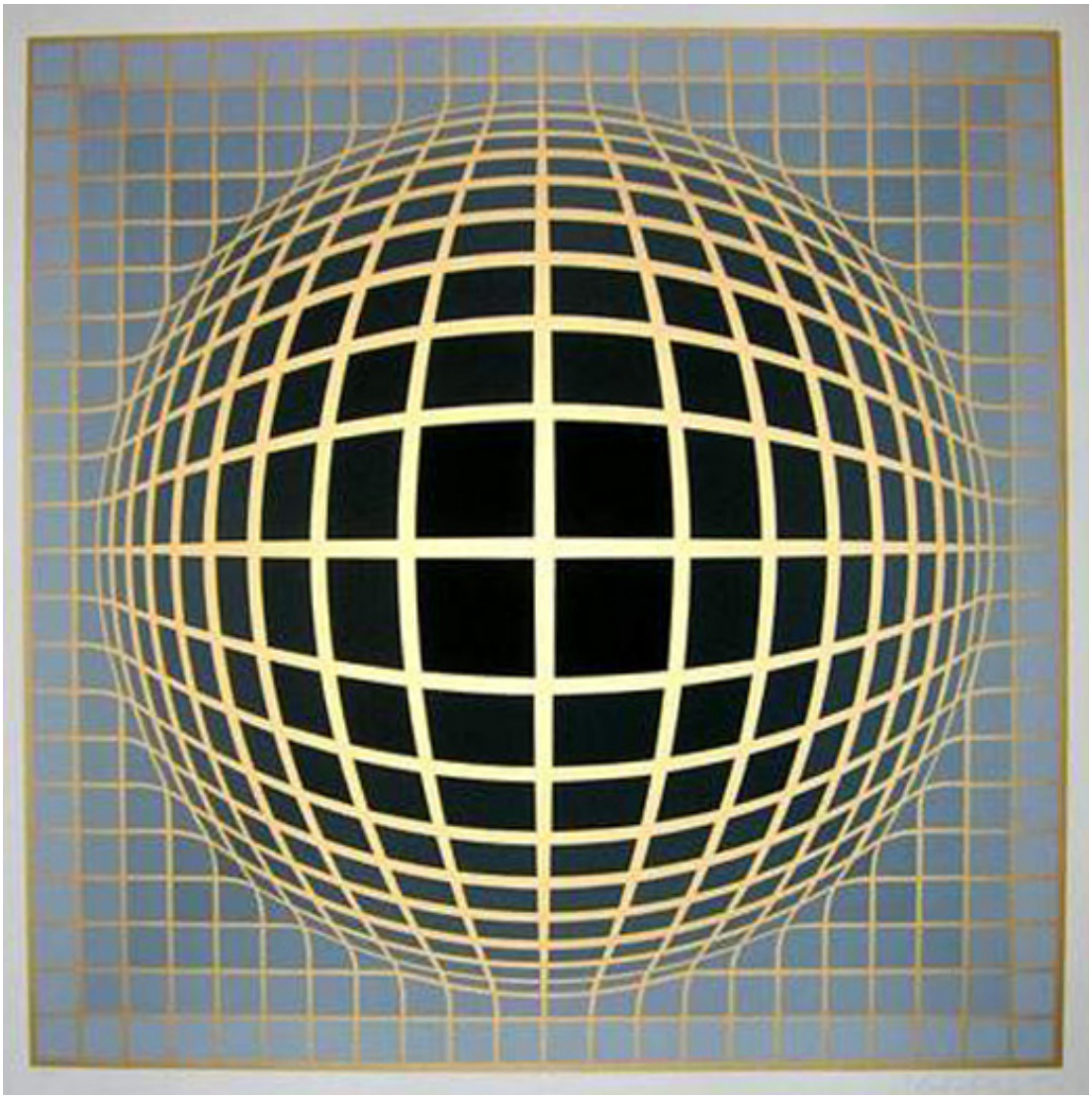
I reserve my warmest gratitude to my wife, Ashley, for willingly enduring, in good humour, the process of research that is realized in this thesis. I am also forever indebted to my parents, Elizabeth and Philip, for their continuous encouragement.

I would like to thank my supervisor Simon Prince without whom this thesis would not have been possible. Simon's clarity of thought and ability to piece together the subject of computer vision in a fathomable way has been invaluable to my endeavor.

The staff and students in the Department of Computer Science at UCL have made my doctoral studies a fulfilling experience, including: Danny Alexander, Simon Arridge, Gabriel Brostow, Bernard Buxton, Lewis Griffin, Simon Julier, Jan Kautz, Min H. Kim, Martin Parsley, Chris Sennanayake, Tony Shepherd, Anthony Steed, James Tompkin and Sara Vicente. A special mention must go to founding members of the Prince Lab, Jania Aghajanian, Yun Fu, Peng Li, Yoshiki Mizukami, Umar Mohammed and Jonathan Warrell who have diligently co-authored, read draft papers and provided constructive criticism and endless practical support throughout my studies.

I would also like to thank practitioners in the wider community for discussions that helped influence the direction of my research. Special thanks must go to Alexander Berg, Richard Bowden, Daniel Cremers, Alexi Efros, Francisco J. Estrada, Mark Everingham, Andrew Fitzgibbon, Leo Grady, Peter Hall, Vladimir Kolmogorov, Mikheal Sisinsev, John Winn and Andrew Zisserman.

Lastly, I would like to praise the organizing committee of PASCAL Visual Object Challenge. Most working scientists depend on good data to make progress; Tycho Brahe's diligent astronomical observations made possible Johannes Kepler's new theories of astronomy. I believe that successful theories for vision will depend on large quantities of data – and whilst I think the term "Labeling Party" stretches the practical use of the word "party" – I found the experience surprisingly rewarding.



SPHERE IN RELIEF, by Victor Vasarely, circa 1975. Original serigraph in different tones of silver and gold.

Human perceptual experience integrates relationships between spaces, objects and sampling structures. This has been exploited by artists for aesthetic merit, including this example from the Op Art movement of the mid 20^{th} century.

Contents

1 Introduction **17**

1.1 Vision Problems 17

1.2 Motivating new segmentation methods 20

1.3 Summary of Contributions 23

1.4 Structure of Thesis 23

1.5 Publications 24

2 Literature Review **25**

2.1 Introduction 25

2.2 Segmentation 27

2.2.1 Agglomerative Algorithms 27

2.2.2 Divisive Algorithms 31

2.2.3 Partitional Algorithms 34

2.3 Labelling 36

2.3.1 Random Field Models 37

2.3.2 Potential functions 39

2.4 Applications of Superpixels 42

2.5 Datasets and Evaluation 49

2.5.1 Datasets 49

2.5.2 Evaluation Methodology 54

2.5.3 Evaluation of Boundary Detectors 57

2.6 Uses of Segmentation and Labelling 57

2.7 Conclusions 61

3 Superpixel Lattices **63**

3.1 Introduction . . . 63

3.1.1 Superpixels and the role of over-segmentation . . . 64

3.1.2 Motivating a new algorithm . . . 66

3.2 Greedy Regular Lattices . . . 71

3.2.1 Overview of our approach . . . 71

3.2.2 Minimum Cost Paths . . . 72

3.2.3 Directed Acyclic Graph . . . 74

3.2.4 Directed Grid Graph . . . 77

3.2.5 Path constraints . . . 78

3.3 Parameters . . . 81

3.4 Qualitative Evaluation . . . 88

3.4.1 Consistent pixel positions . . . 88

3.4.2 Consistent spatial relations . . . 89

3.4.3 Scale Hierarchy . . . 89

3.4.4 Compactness . . . 89

3.4.5 Graph Isomorphism . . . 90

3.5 Quantitative Evaluation . . . 90

3.5.1 Comparison of competing graphs . . . 90

3.5.2 Evaluation methodology . . . 94

3.5.3 Comparing Boundary Maps . . . 97

3.5.4 Comparison to other algorithms . . . 99

3.6 Runtime and Computational Complexity . . . 110

3.7 Connectivity . . . 113

3.8 Merging Superpixels . . . 115

3.9 Failure Modes . . . 118

3.10 Temporal Stability . . . 124

3.11 Discussion and Future Work . . . 126

3.11.1 Relation to other work . . . 127

3.11.2 Evaluation methodology . . . 128

3.11.3 Limitations of the algorithm . . . 129

3.12 Conclusions . . . 130

4 Scene Shape Priors **131**

4.1 Introduction . . . 131

4.2 Distribution of Ground Truth Regions . . . 133

4.2.1 Scene Shapes . . . 136

4.3 Adaptive Regular Lattices . . . 137

4.3.1 Non-uniform Strips . . . 138

4.3.2 Non-uniform Minimum Cost Path Algorithm . . . 139

4.3.3 Scene shape warping . . . 140

4.4 Boundary distribution prior . . . 146

4.4.1 Learning . . . 147

4.4.2 Inference . . . 150

4.5 Quantitative Evaluation . . . 150

4.5.1 Selecting clusters . . . 151

4.5.2 Comparison to GRL algorithm . . . 152

4.5.3 Comparison to other algorithms . . . 154

4.6 Discussion and Future Work . . . 155

4.6.1 Comparison to existing methods . . . 158

4.6.2 Evaluation . . . 159

4.6.3 Limitations of the algorithm . . . 162

4.7 Conclusions . . . 162

5 Lattice Cut **164**

5.1 Introduction . . . 164

5.2 Overview of our approach . . . 166

5.3 Multi-label MRFs and Constraint Edges . . . 167

5.4 Graph construction for ordered layers . . . 170

5.4.1 Full graph construction . . . 173

5.5 Lattice Cut . . . 176

5.5.1 Model Grid . . . 177

5.5.2 Parameters . . . 178

5.6 Qualitative Evaluation . . . 179

5.7 Quantitative Evaluation . . . 180

5.7.1 Potential terms . . . 181
5.7.2 Comparing Boundary Maps . . . 181
5.7.3 Comparison to other algorithms . . . 181
5.8 Runtime and Computational Complexity . . . 182
5.9 Failure Modes . . . 190
5.10 Discussion and Future Work . . . 191
5.10.1 Comparison to existing methods . . . 191
5.10.2 Initialization . . . 194
5.10.3 Graph construction and higher order terms . . . 195
5.10.4 Multiple Segmentations . . . 196
5.10.5 Efficiency . . . 196
5.10.6 Potential functions . . . 196
5.11 Conclusions . . . 197

6 Conclusion **199**
6.1 Summary of Findings . . . 199
6.2 Limitations and Future Work . . . 200
6.3 Final Remarks . . . 202

Appendices **204**

A Path constraints **204**
A.1 Constraints for G_{dag} . . . 209
A.2 Constraints for G_{gg+} . . . 210

B Tables of Results **214**
B.1 Chapter 3. Berkeley Segmentation Database . . . 215
B.2 Chapter 3. Pascal Visual Object Classes . . . 216
B.3 Chapter 3. Cambridge Labeled Video Database . . . 217
B.4 Chapter 4. Cambridge Labeled Video Database . . . 218
B.5 Chapter 4. Cambridge Labeled Video Database . . . 219
B.6 Chapter 5. Berkeley Segmentation Database . . . 220
B.7 Chapter 5. Pascal Visual Object Classes . . . 221
B.8 Chapter 5. Cambridge Labeled Video Database . . . 222

B.9 Chapter 5. Cambridge Labeled Video Database 223

C Performance Measures **224**

C.1 Region Measures 224

C.1.1 Cover Score 227

C.1.2 Global Consistency Error 228

C.1.3 Rand Index 228

C.1.4 Variation of Information 229

C.1.5 Combined Segmentation and Detection - F_{sd} 230

Bibliography **232**

List of Figures

1.1 Modes of variation in image formation. 18
1.2 Ambiguity of local image patch. 19
1.3 “Photo Pop-up” pipeline 22

2.1 Agglomerative clustering 28
2.2 Single-linkage segmentation 29
2.3 Density Estimation 30
2.4 Graph partitioning 32
2.5 Mixture models 34
2.6 Random fields models 37
2.7 Graph Cuts 39
2.8 Pb feature responses 40
2.9 Example boundary detectors 41
2.10 Superpixels and CRFs 48
2.11 Dataset exemplars. 50
2.12 Class, Instance and Region ground truth. 52
2.13 Segmentation vs. Detection 56
2.14 Precision Recall curves for CamVid and BSD. 58
2.15 Exploitation 59

3.1 Illustration of learned regional *label features*. 66
3.2 Representing images as graphs. 68
3.3 Problems of varying topology. 68
3.4 Incremental construction of superpixel lattice. 71
3.5 Estimation of optimal path through an image strip 73
3.6 Graph construction for directed acyclic graph 76

3.7 Graph construction for directed grid graph . . . 79
3.8 Example Path Crossings. . . . 80
3.9 Examples parameter settings of a greedy regular lattice. . . . 82
3.10 Tortuosity. . . . 86
3.11 Exploring lattice parameters. . . . 87
3.12 Scale Hierarchy. . . . 89
3.13 Comparing superpixel properties. . . . 91
3.14 BSD11 dataset . . . 92
3.15 Explained variation . . . 93
3.16 Robust interpolation . . . 95
3.17 Algorithm ranking using Borda score . . . 95
3.18 Comparing Boundary Maps . . . 98
3.19 Algorithms vs Metrics . . . 100
3.20 Selected performance measures on BSD . . . 102
3.21 BSD - High/Mid/Low ranked examples. . . . 103
3.22 BSD - High/Mid/Low ranked examples. . . . 104
3.23 VOC - High/Mid/Low ranked examples. . . . 106
3.24 VOC - High/Mid/Low ranked examples. . . . 107
3.25 CamVid - High/Mid/Low ranked examples. . . . 108
3.26 CamVid - High/Mid/Low ranked examples. . . . 109
3.27 Comparison of algorithm runtime . . . 111
3.28 Region adjacency graphs . . . 114
3.29 Connectivity . . . 114
3.30 Merged lattices . . . 116
3.31 Merging lattices with the Mean Shift algorithm . . . 116
3.32 Merged Lattice - High/Mid/Low ranked examples. . . . 119
3.33 Cross-over error. . . . 120
3.34 Close-off error. . . . 121
3.35 Bo-tie error. . . . 121
3.36 Strip error. . . . 121
3.37 Diagonal error. . . . 122
3.38 A/Y-like error. . . . 122

3.39 Tortuosity error. . . . 122
3.40 Failure Modes . . . 123
3.41 A seeded 3D lattice. . . . 125
3.42 Superpixel Stability. . . . 125

4.1 Missing cars example. . . . 132
4.2 Size and distribution of dataset ground truth regions 134
4.3 Size and distribution of CamVid dataset ground truth regions 135
4.4 Prototypical views from CamVid dataset. . . . 135
4.5 Influence of perspectives. . . . 138
4.6 Construction of non-uniform strip. . . . 139
4.7 Warping minimum cost paths. . . . 141
4.8 Cumulative Image Co-ordinates. . . . 142
4.9 Interpolating warped data points. . . . 143
4.10 Warping Boundary Distribution Image 144
4.11 Comparison of warped images. . . . 145
4.12 Clustered Latent Trait Model 147
4.13 Learned cluster means. . . . 148
4.14 Sampling from one cluster during inference. . . . 150
4.15 Foveation of warped lattices. . . . 152
4.16 Comparing GRL and ARL 153
4.17 Scene shape prior captures distant objects. . . . 154
4.18 Plots of mean F_s against mean F_d 156
4.19 Box-whisker plots using F_{sd} 159
4.20 CamVid - High/Mid/Low ranked examples. . . . 160
4.21 Size and distribution of CamVid ground truth before and after warping . 161

5.1 GrabCut . . . 165
5.2 Lattice Cut. . . . 168
5.3 Label constraints for ordered layers 169
5.4 Example graph construction for ordered layers 171
5.5 Imposing sequential and order constraints 172
5.6 Enforcing all layers. . . . 173

5.7 Full ordered layer graph construction. 175
5.8 Toy multi-label example comparing α expansion and ordered layers . . 175
5.9 Lattice constraints on orthogonal MRFs. 176
5.10 Model schedule. 180
5.11 Iterative convergence of Lattice Cut. 182
5.12 CamVid - High/Mid/Low ranked examples. 183
5.13 CamVid - High/Mid/Low ranked examples. 184
5.14 CamVid - High/Mid/Low ranked examples. 185
5.15 Comparing Boundary Maps 186
5.16 Comparing GRL and LC 187
5.17 Comparison of algorithm runtime 188
5.18 Shrinkage error. 191
5.19 Porous error. 191
5.20 Failure Modes 192

6.1 Lattice Improvement. 201

A.1 Example Path Crossings. 205
A.2 Parallel path constraints for G_{dag}. 207
A.3 Orthogonal path constraints for G_{dag}. 208
A.4 Orthogonal path constraints for G_{gg+}. 212

C.1 Binary Confusion Matrix. 225
C.2 Cover Score 226
C.3 Global Consistency Error 227
C.4 Rand Index 229
C.5 Detection with over-segmentation. 230
C.6 F_{sd} 231

List of Tables

1.1 Common Vision Tasks 21

2.1 Applications of superpixel algorithms in the vision processing pipeline . 45

3.1 Pixel vs Superpixel properties 70
3.2 Number of superpixels in applications 84
3.3 Performance of different graph constructions on BSD11. 92
3.4 Boundary map ranking. 97
3.5 Algorithm ranking. 99
3.6 Greedy Lattices with human labeled ground truth 117
3.7 Merging superpixels 120

4.1 Selecting cluster. 151
4.2 Algorithm ranking. 156
4.3 Results of F_{sd} by class for competing algorithms at ~ 600 superpixels. . 157
4.4 GRL-MS vs ARL-MS for selected CamVid classes using F_{sd}. 158

5.1 Potential terms. 181
5.2 Boundary map ranking. 189
5.3 Algorithm ranking. 190

B.1 Chapter 3. Performance of GRL algorithm on BSD. 215
B.2 Chapter 3. Performance of GRL algorithm on VOC. 216
B.3 Chapter 3. Performance GRL algorithm on CamVid. 217
B.4 Chapter 3. Performance ARL algorithm on CamVid. 218
B.5 Chapter 3. Performance measured using F_{sd} on CamVid for selected Classes. 219

B.6 Chapter 3. Performance of LC algorithm on BSD. 220
B.7 Chapter 3. Performance of LC algorithm on VOC. 221
B.8 Chapter 3. Performance LC algorithm on CamVid. 222
B.9 Chapter 3. Performance measured using F_{sd} on CamVid for selected Classes. 223

Chapter 1

Introduction

In this chapter we introduce vision tasks that make use of small image segments called superpixels. We argue that superpixels are useful and this approach motivates the work in subsequent chapters.

1.1 Vision Problems

Interpreting images effortlessly is something most humans take for granted. However, the difficulty of general vision problems can be appreciated if we consider the modes of variation that are present in a common scene. For example, consider the scene in Figure 1.1. We begin with a unknown projection of an unknown 3D world. If we know that it is a road scene and that we are an observer on the ground plane we can often assume something about the geometry and depth (see Figure 1.1b). Moreover, we might expect to find a certain number of 'road classes' (i.e.cars or pedestrians - not bananas) and make predictions about their likely ordering and distribution across the image. We also need to think about the properties of each object: particular instance appearance, orientation, articulation, layout occlusions and so forth. Finally, we need to consider the capture conditions of our camera system including lighting, shading, shadowing, reflections, atmospheric conditions and camera effects. The full image, Figure 1.1h, is a combination of all these different modes of variation and teasing them apart is currently a very challenging task.

Furthermore, a small set of pixels considered in isolation seldom looks much like anything at all [119]. Therefore interpreting the signal in an image patch relies on inference about several different aspects of the scene. One illustration of this can be seen in Figure 1.2 where it is easy to interpret a 30×10 pixel image patch as a cy-

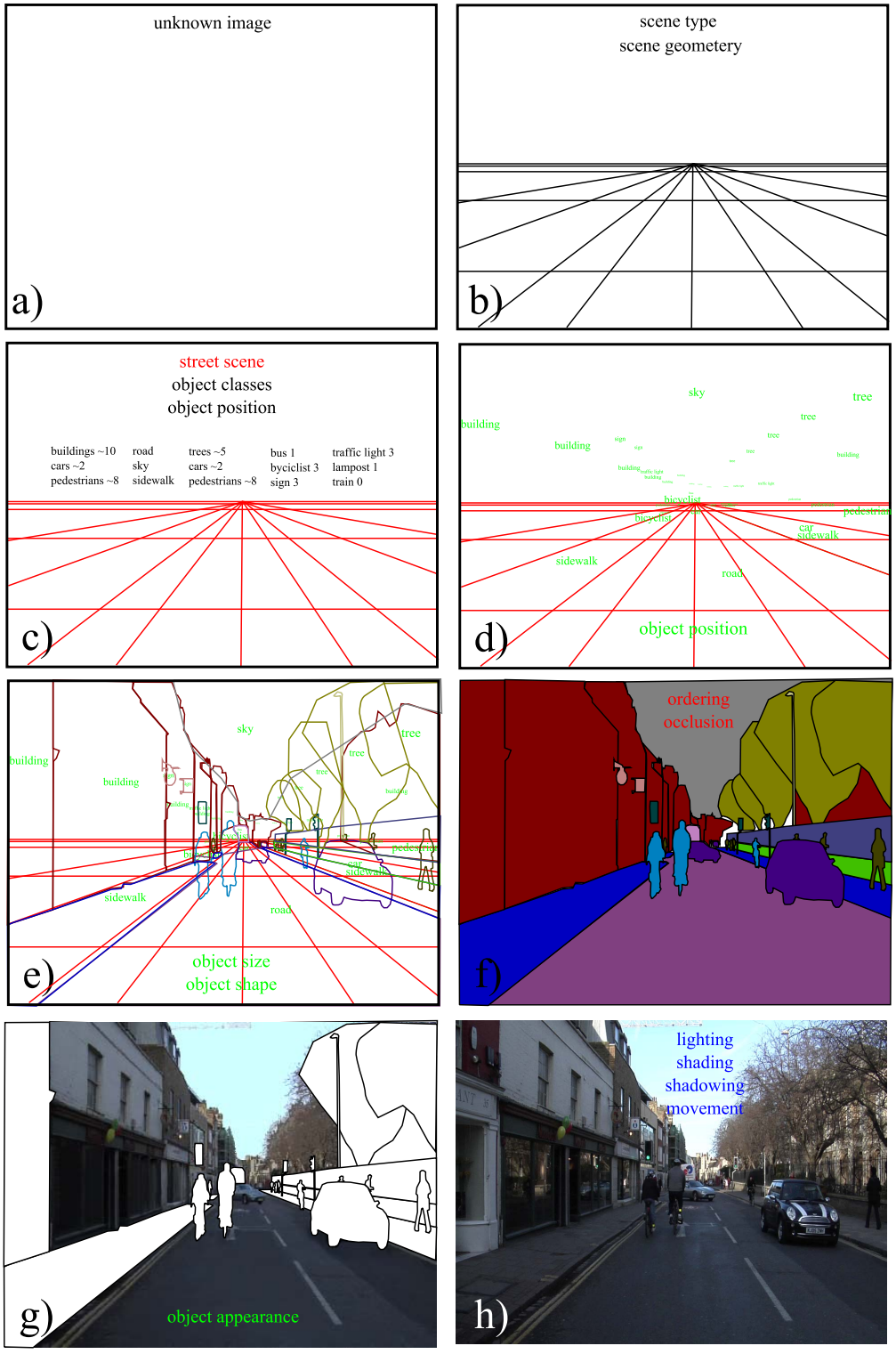


Figure 1.1: Modes of variation in image formation. a) Unknown projection of unknown 3D world b) Scene category, geometry and depth. c) Presence of object classes. d) Estimates of object position. e) Object size, shape and orientation. f) Ordering and occlusions of objects in scene. g) Object appearance. h)Image capture, lighting, shading, shadows and camera effects. Original image taken from the CamVid dataset [39]. Image sequence inspired by Winn [281].

a)

b)

Figure 1.2: Ambiguity of local image patch in street scene. a) Original image doctored to replicate an image patch in region where it is difficult to detect structural differences. This is intended to demonstrate that there is more to interpreting image data than correct classification of local windows. b) 'Cyclist' pixels appear in building region on the left and on the road region, image right, where it is easier to interpret them as a cyclist on the street. Images inspired by the work of Torralba [260] and Hoiem et al. [119].

clist in one part of the image, when situated on a road between between cars, but as signage on a building, when floating on the left hand side over shop fronts. Modeling the relationships between different modes of variation makes vision problems in unconstrained environments challenging. It is currently an open research problem as to how to estimate the many different sources of variability [23, 120].

Early successful applications of computer vision made use of 'constrained' capture conditions to control or limit the effect of different modes of variation. Examples include using machine inspection on the conveyor belts [62] or on motion tracking on film set with known backgrounds and markers [102]. This divide and conquer approach has also lead to the separation of vision problems into tasks with more limited scope. A list of common vision tasks can be seen in Table 1.1. Two tasks that are the focus of the thesis, *segmentation* and *labelling*, are introduced in more detail in Chapter 2.

1.2 Motivating new segmentation methods

Focusing research effort on restricted vision tasks has resulted in some dramatic successes. Despite the previous claim - that a small set of pixels considered in isolation seldom looks much like anything at all - there are several notable exceptions. For example, the detection of the *frontal faces* class is a very restricted problem that has yielded to existing methods [232, 270] and it is now common for this technology to be found in consumer electronics [10]. Frontal face detection methods commonly exploit a rectangular region of support called a *sliding window* [232, 270, 61]. State-of-the-art methods for scene understanding continue to exploit the sliding window representation to improve performance on certain classes [149].

However, the success of methods that exploit a rectangular region of support vary considerably by class [172]. For instance, the results on the PASCAL VOC Challenge [87] demonstrate a difference in performance for man-made structured objects (car, bicycle, bus) from articulated soft natural objects (sheep, dog, cow). Moreover, ranking results on the challenge dataset show that high ranked positive images are those where there is a large single object of interest in a canonical pose compared to low ranked images that exhibit objects at small scales (ambiguous at the scale of a patch window) or objects that are heavily occluded [87].

Limitations of the sliding window representation have led to methods that use

Vision Task	Description	Example appraoches
Recognition	*Identity* recognition is the task of selecting an individual from a population of similar instances by modeling within class variation.	Turk and Pentland 1991 [262] Belhumeur *et al.* 1998 [26] Prince and Elder 2007 [210] Aghajanian and Prince 2008 [13]
Regression	Estimating a continuous attribute of an individual, eg. age, height, orientation.	Okada & Soatto 2008 [197] Aghajanian *et al.* 2009 [14] Navaratnam *et al.* 2007 [193]
Generation	Creating entirely novel image data having learned a model of a particular type of variation.	Efros & Leung 1999 [81] Efros and Freeman 2001 [80] Mohammed *et al.* 2009 [181]
Classification	Associating image data with a particular set, or class, of individuals. This might refer to a whole image or region within an image.	Shotton *et al.* 2006 [239] Dollar *et al.* 2006 [74] Hoiem *et al.* 2007 [124] Gould *et al.* 2008 [105]
Detection	Finding a sub-image, window or bounding box, of an object of interest.	Scheiderman & Kanade 2000 [232] Viola & Jones 2001 [270]
Labeling	The task is to associate a particular label from a set $\|\mathcal{L}\|$ to every pixel in an image. This approach has also been used to produce a solution to *Segmentation* problems (below).	He *et al.* 2004 [115] Rother *et al.* 2004 [225] Agarwala *et al.* 2004 [12] Kohli 2008 [138] Warrell *et al.* 2009 [275]
Segmentation	Dividing an image into a disjoint set of pixel regions - often a pre-processing step. See Table 2.1	Shi & Malik 2000 [238] Boykov & Jolly 2001 [35] Comanicui & Meer 2002 [53]

Table 1.1: Some common vision tasks. A short description of different vision tasks and example approaches. Each one of these tasks has many different approaches and each its own important literature. The number of examples do not represent the breadth of study in each separate task. Approaches to the Segmentation and Labeling tasks are discussed in greater detail in Chapter 2

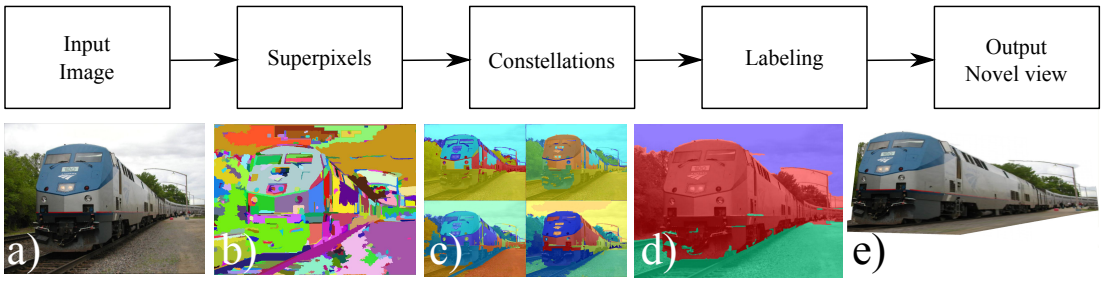


Figure 1.3: "Photo Pop-up" pipeline. Example vision pipeline that exploits superpixels as an early pre-processing step to be used as the basis for inference at subsequent stages. Reproduced from [118].

data-dependent spatial support for vision algorithms [172]. Current vision systems often exploit a disjoint sets of pixels - commonly referred to as a *segmentation*. Figure 1.3 illustrates one example of a vision pipeline that uses the output of a segmentation algorithm as the basis for subsequent processing steps [119]. Importantly, Malisiewicz and Efros [172] demonstrate that better spatial support, provided by ground truth segmentations, improves the performance of classifiers over sliding window approaches. A full discussion of these issues is presented at the beginning of Chapter 3.

There is now a large and growing body of work [238, 218, 89, 172, 159, 55, 268, 11] that focuses on segmentation techniques whose output can be used within automated vision pipelines similar to that illustrated in Figure 1.3. Segments that are used in this context are commonly referred to as *superpixels*. Examples of superpixel methods and their applications are presented in Chapter 2.

Unfortunately, current methods suffer from various drawbacks. For example, they can be slow to compute [238, 185] or susceptible to noisy estimates of data [89]. Recognizing this limitation, several recent approaches have used multiple superpixel segmentations as a set of hypotheses about spatial support [118, 227, 119] or integrated them to produce robust regions of support based on label consistency [138, 148]. Despite recent progress, there is currently no dominant algorithm.

Moreover, it is interesting to observe that superpixel methods seldom retain the useful properties of the pixel representation that they build upon, see Table 3.1. If the superpixel is to become a standard representation of image data capable of facilitating additional vision tasks then further research is required.

1.3 Summary of Contributions

In the thesis we propose a new approach to superpixel algorithms. Our goal is to develop computational approaches that are broadly useful, in the way that other low-level techniques such as edge detection [43] are used in a wide range of computer vision tasks. The primary contributions of this thesis are presented in the following chapters:

Chapter 3 - In this chapter we discuss the motivations for superpixels in more detail. We argue that a number of desirable properties of pixels should be maintained by superpixels and that this is not possible with existing algorithms. We develop an algorithm that satisfies some of these additional properties and demonstrate performance on standard datasets.

Chapter 4 - In this chapter we highlight that the superpixel lattice, and over-segmentation algorithms in general, have noticeable limitations on certain datasets. To overcome this we introduce an algorithm that learns the empirical distribution of boundaries in a supervised learning paradigm and use this to improve the segmentation.

Chapter 5 - In this chapter we improve upon the algorithm introduced in Chapter 3. We reformulate the superpixel lattice algorithm based on a novel Markov random field framework it so that it is possible to give global guarantees on the solution.

1.4 Structure of Thesis

The thesis is divided into seven chapters, the first of which is this introduction, the second is a literature review. Chapters 3-5 develop methods for producing a representation for pixel data with a fixed structure or topology - the superpixel lattice. Limitations and directions of future work are discussed in Chapter 6. Finally, there are a set of appendices. Appendix A, gives details of implementations. Appendix B sets out tables of quantitative results in full. Appendix C sets out the performance measures used during the quantitative evaluation of algorithms in the thesis.

1.5 Publications

Work contributing to the methods presented in the thesis have been published in the following conference papers:

Chapter 3: Alastair P. Moore, Simon J. D. Prince, Jonathan Warrell, Umar Mohammed and Graham Jones, "Superpixel Lattices", In *Proceedings of the Computer Vision and Pattern Recognition* (CVPR), 2008.

Chapter 4: Alastair P. Moore, Simon J. D. Prince, Jonathan Warrell, Umar Mohammed and Graham Jones, "Scene Shape Priors for Superpixel Segmentation", In *Proceedings of the International Conference on Computer Vision* (ICCV), 2009.

Chapter 5: Alastair P. Moore, Simon J. D. Prince and Jonathan Warrell, "Lattice Cut - Constructing superpixels using layer constraints", In *Proceedings of the Computer Vision and Pattern Recognition*, (CVPR) 2010.

Chapter 2

Literature Review

This chapter sets out to review important literature that relates to the subject of the thesis. The work addresses two principal themes: Segmentation and Labeling.

2.1 Introduction

There are several alternative approaches to segmentation tasks. In general there are two questions to be answered: First, is there a single correct segmentation? There may be several possible interpretations given prior world knowledge. Secondly, how should we specify the prior world knowledge? Some prior knowledge is low-level such as colour or textural uniformity, some is mid-level such as gestalt cues of similarity, proximity and symmetry and finally some is high-level like object models or scene layout.

It was noted very early in computer vision research that the segmentation task itself is difficult [196, 254] and this has motivated several different approaches. There is a considerable body of work that focuses on using human input, called *supervised* segmentation. For example, a user can specify an ordered set of pixels on object boundaries [187, 280]. More recent approaches include a user specifying *seeds* which are pixels that belong to one of several object classes [35, 108]. Alternatively, it is possible to use object models to provide high-level information to limit the space of possible solutions. Providing limited information about a likely segmentation is often referred to as *semi-supervised* segmentation and the exploitation of object models a *top-down* approach. One example is where a user initializes contours that represent a model of an object in the image. This type of interaction incorporates a large family of methods including active contours [132, 30] and level sets [234].

One consideration that influences the design of segmentation algorithms is

whether the goal is to produce a final "correct" segmentation or alternatively to arrive at a set of useful image primitives to guide further image reasoning. Intermediate representations of an image are referred to as *under-segmentations* or *over-segmentations* depending on whether they are likely to combine separate objects in an image within one segment or divide an object into multiple segments. An approach based solely on the low-level coherence of brightness, colour, texture or motion cues it is commonly referred to as a *bottom-up* approach and the absence of external input is called *unsupervised* segmentation.

Image primitives based on a bottom-up approach that over-segment an image have colloquially become known as "superpixels". Often the goal of this representation is to break an image into small pieces so that there is a high probability that each piece belongs to only one object in the image. The thesis addresses the sub-goal of many vision tasks: unsupervised over-segmentation - creating superpixels. The motivations for this approach are discussed in greater detail at the beginning of Chapter 3.

The review focuses on algorithms for bottom-up segmentation or methods that build on this approach. It is presented in four parts: Segmentation, Labeling, Applications and, Evaluation and Datasets. *Segmentation*, Section 2.2, sets out the tool-box of techniques that have been applied to the over-segmentation problem. The section categorizes existing methods and reviews several core algorithms behind each approach. One particular approach, *Labeling*, has shown great promise in many vision applications, including supervised segmentation [35, 141]. Work presented in Chapter 5 makes use of a labelling approach in an energy minimization framework and we therefore introduce this separately in Section 2.3. The section also introduces notation used later in the thesis.

Applications of Superpixels, Section 2.4, gives examples where the methods just explained have been used to tackle different vision tasks. The section highlights some of the difficulties and limitations of using existing techniques. The datasets and evaluation methodology used for comparing the performance of algorithms presented in the thesis are introduced in Section 2.5. Finally, in Section 2.6 we give an overview of some current applications of segmentation methods.

2.2 Segmentation

Segmentation is a type of *clustering* problem and work on clustering points in the analysis of datasets is an extensive area of research with applications that extend well beyond computer vision. A recent review that puts developments of clustering methodology in historical context is [130].

In general there are two representations of data in segmentation problems: segmentation of the image plane and segmentation of a feature space [175, 291]. By *feature space* we mean a description of the simple pixel data. For instance, a simple example would be a feature vector containing red, blue and green pixel data with the x and y image co-ordinates [53]. We shall use the term data point, pixel, feature point and graph node interchangeably in this section and assume that techniques apply equally to each representation or that the distinction is clearly given. There are three questions that need to be answered to construct a segmentation algorithm. 1. What measure should be used for similarity between data points? 2. What criterion can be used to identify a good clustering of the data? 3. How can the clustering be computed efficiently? The first question is discussed at greater length in Section 2.3.2. The answers to the last two questions divide segmentation methods into three broad clustering techniques - *agglomerative*, *divisive* and *partitional* [130].

Both agglomerative and divisive methods naturally lead to hierarchies of clusters. An *agglomerative* approach begins with each data point as its own cluster and iteratively merges clusters. Alternatively, a *divisive* approach begins with one cluster and recursively divides its constituent members into into two or more new clusters. In contrast *partitional* clustering algorithms can be interpreted as assigning data points to all clusters simultaneously, given the implementation. Partitional approaches tend to be characterized by fitting a fixed set of models to data rather than evaluating a measure of similarity between clusters. In the following sections we illustrate each approach with several different algorithms.

2.2.1 Agglomerative Algorithms

Perhaps the earliest work on clustering in images is that of Nagao et al. [191]. In this work the distance between grey level values is substituted for the distance in a single-linkage clustering paradigm [106]. The algorithm begins by taking a set of data points

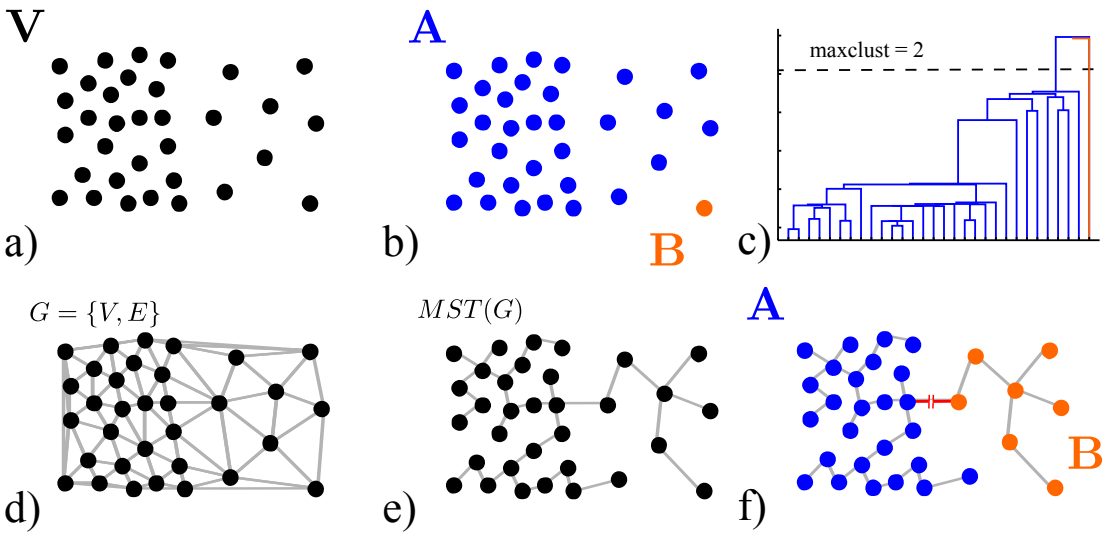


Figure 2.1: Agglomerative clustering. a) A set of feature points, V b) Separation into two clusters using a single-linkage criterion. An unbalanced clustering is found. c) Dendrogram of full single-linkage clustering of points in [a], where a threshold has been set to two clusters seen in [b] . d) Planar graph representation of points in [a] based on Delaunay triangulation [64]. e) Minimium spanning tree (MST) of the graph in [d]. Clustering based on setting merging criterion on the MST.

and merges them based on the minimum distance between points.

However, a common problem with single-linkage is that it can produce a "chaining effect" which results in long elongated clusters or alternatively many small compact clusters [78]. This means that results are very sensitive to noise or initial conditions and segmentations tend to be unstable.

If the single-linkage algorithm is allowed to merge all nodes into one cluster the results is the *minimum spanning tree* (MST) of a fully connected graph. This is illustrated in Figure 2.1. The shortest distance hierarchy is the core of the greedy algorithm by Kruskal [142] for solving the minimum spanning tree problem and the basis of the dendrogram representation of the single linkage clustering (see Figure 2.1c). Early work on using the MST for segmentation includes the work of Zahn [292].

A connected component of an undirected graph is a subgraph where any two vertices are connected to each other with a path. Placing a fixed threshold on edges in the spanning tree can break it into a spanning forest - such that each connected component of the graph is a minimum spanning tree. However, this approach suffers from the same problems as single-linkage. An example of this in segmentation is illustrated

Figure 2.2: Single-linkage segmentation. a) Original image with three perceptually distinct regions. Reproduced from [89]. b) Single-linkage segmentation merges uniform region and ramp edge and divides high variance region. c) Result of the FH algorithm.

in Figure 2.2. We can see that minimum distance linkage can merge uniform regions and ramp edges and divide regions of high variance into several different clusters (see Figure 2.2b).

Alternative linkage schemes, including average-linkage [243, 190] and normalized-linkage [265], suffer from problems based on fixed thresholds. However, one linkage approach that overcomes the difficulties illustrated in Figure 2.2a is the algorithm of Felzenszwalb and Huttenlocher [88]. The authors propose a modification of Kruskal's algorithm that evaluates a variable region predicate for deciding the evidence of a link between clusters. The predicate they propose for a region is "not too course and not too fine". This is based on two terms that evaluate the *internal difference* of a component with the *difference between* components. The *internal difference* is the largest weight in the minimum spanning tree of the component and the *difference between* components is the minimum weighted edge connecting two components.

The authors prove that their predicate guarantees certain global properties on the segmentation, that it is very efficient and that modification to the *difference* measure renders the problem NP-hard. Their approach produces useful segmentations and has become popular in image segmentation schemes (see Table 2.1). We use this algorithm in the quantitative evaluation in the thesis and refer to it with the acronym `FH`.

Despite the global guarantees on the granularity of the segmentation the `FH` algorithm still suffers form the inherent stability issues that result from the use of MST. This is demonstrated in Section 3.10. Additionally, it is a challenge to maintain *compactness* of the superpixels. Compactness refers to regions being small, compact, with a quasi-uniform distribution over the image [159] and is discussed further in Chapter 3.

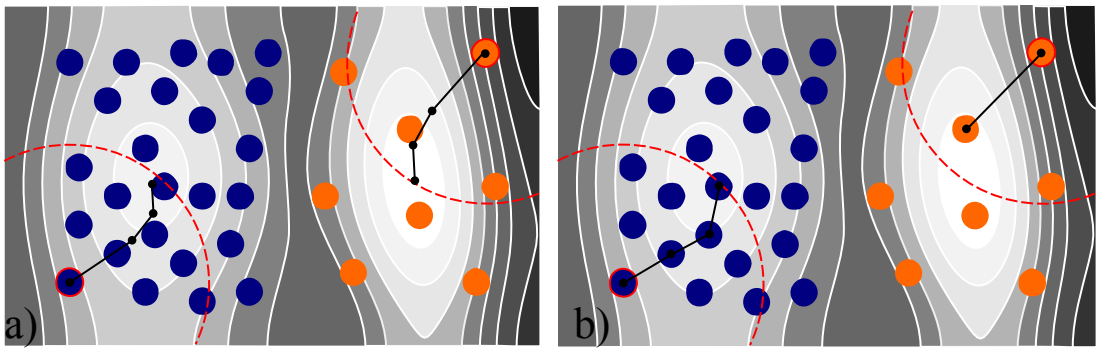


Figure 2.3: Density Estimation. a) Example of Mean Shift iteration. Starting from original point (circled red) the data point takes a new value based on the mean of data points within a fixed radius. Subsequent iteration result in a trajectory for each data-point towards a local mode b) Example of Mediod Shift interaction. At each iteration the updated position of the data point must be a member of the original set. Blue and orange points represent two local "basins of attraction".

Several alternative approaches based on the idea of connected components using a spanning forest are present in the literature [242]. One very early approach that continues to receive attention [55, 56] is the work on watersheds [71, 269]. This approach begins by considering the image as a topographic surface where the dark and light regions of the image correspond to valleys and ridges. Intuitively, the algorithm can be considered to divide the surface into regions based on "catchment basins" of water. Recent analysis demonstrates the theoretical relationship between the watershed approach an other graph based energy minimization algorithms [241, 60]. However, watershed results can suffer from leaks (chaining effects) and unlike the method we propose in the following chapters, there are only weak guarantees on the the topology of the watershed output [57].

The intuitive idea of "basins of attraction" is also exploited in approaches that find dominant modes in a feature space representation. One recent approach that has proved useful in many vision applications is the Mean Shift algorithm [98, 53]. Mean Shift (MS) is based on non parametric density estimation where the rationale is that dense regions in the feature space correspond to local maxima of an underlying probability distribution.

Figure 2.3a illustrates the algorithm. At each iteration the Mean Shift procedure entails that a data point moves towards the local mean of data points within a given

threshold (radius). Successive iterations result in feature points migrating toward a local mode of the feature space. It has proved very popular in recent vision literature both for speed, ease of implementation and the relatively few number of parameters. Implementations of these algorithms tend to merge data points whose trajectories converge within a given threshold, to increase efficiency, which serves to highlight the similarity with other agglomerative methods. However, the algorithm can result in fragmentation of constant gradient regions.

The `MS` algorithm is related to kernel density estimation that is a well studied problem in other fields [50]. Interestingly, it is possible to show that the `FH` algorithm is closely related to the `MS` algorithm with a variable radius of dilation [89], although it should be noted that the public implementation does not use a feature space representation. We use the EDISON implementation of the MS algorithm in our quantitative evaluation [176].

There has been considerable work on producing both efficient, hierarchical and temporal variations [46, 47, 202, 201]. The Mediod Shift algorithm [237] is a variant that uses the most centrally located sample in a set of points rather than the mean, illustrated in Figure 2.3b. This can have advantages when it is difficult to specify a mean of feature vectors. Other recent variations applied to image segmentation problems include the Quick Shift algorithm [267]. Moreover, there are similarities with filtering and estimation approaches, including M-estimators [53, 160, 127], Non-linear diffusion [205, 28], Non-local means [41] and bilateral filtering [259, 207]. However, these methods tend to be used for image restoration and compression artefact removal tasks rather than segmentation.

Mode seeking methods have proved successful in applications, see Table 2.1, and in general these methods tend to be fast. However, these methods are essentially gradient ascent algorithms [45] and are based on the local structure of data. In the next section we present methods that use the global structure of a set of data.

2.2.2 Divisive Algorithms

At the beginning of Section 2.2.1 we saw an *Agglomerative* approach that used a particular graph, the minimum spanning tree, to capture the clustering properties of the data. *Divisive* methods also commonly consider data points as nodes in a weighted

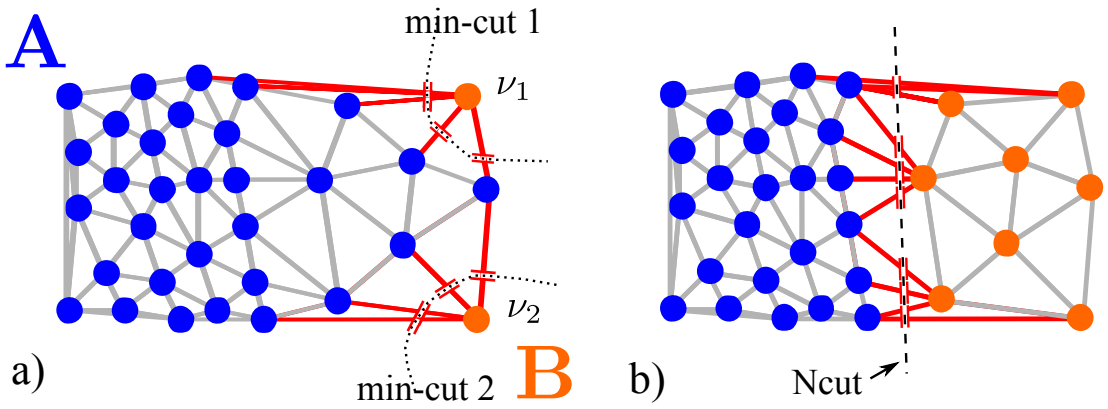


Figure 2.4: Graph Partitioning. a) A set of feature points, $\mathbf{V}$, partitioned into two disjoint sets $\mathbf{A} \cup \mathbf{B} = \mathbf{V}$ using two minimum cuts. Edge weights are inversely proportional to the distance between nodes b) Example using the normalized cut criterion to partition the same graph. Reproduced from [238].

undirected graph $G = \{V, E\}$ where the edge weight is a measure of the similarity between nodes.

Early divisive approaches to image segmentation exploited graph partitions or *cuts* on a graph [292]. A partition of the graph is a separation of the nodes into two disjoint subsets $\mathbf{A}$ and $\mathbf{B}$ where $\mathbf{A} \cup \mathbf{B} = \mathbf{V}$ and $\mathbf{A} \cap \mathbf{B} = \emptyset$. This is illustrated in Figure 2.4. The similarity between sets $\mathbf{A}$ and $\mathbf{B}$ is the sum of the weighted edges removed from the graph G:

$$cut(\mathbf{A}, \mathbf{B}) = \sum_{\mu \in \mathbf{A}, \nu \in \mathbf{B}} w(\mu, \nu) \tag{2.1}$$

This set of weighted edges is referred to as a *cut*. In this paradigm the different properties of the *cut* dictate both the clustering criterion and how it can be efficiently computed.

Some of the earliest papers on using *cuts* for image segmentation use the *minimum cut* [285]. The *minimum cut* is a cut minimizing the sum of the edge weights that belong to the cut. However, this approach met with limited success as the minimum cut criteria is biased toward short boundaries and therefore favors the creation of small sets of isolated regions in a final segmentation. This is illustrated in Figure 2.4a.

Modern approaches to alleviate some of the problems of the minimum cut criterion begins with the work of Shi and Malik [238] who proposed the *normalized cut*. The *normalized cut* criterion measures both the total dissimilarity between sets of nodes $\mathbf{A}$

and $\mathbf{B}$ as well as the total similarity within each set:

$$Ncut(\mathbf{A},\mathbf{B}) = \frac{cut(\mathbf{A},\mathbf{B})}{assoc(\mathbf{A},\mathbf{V})} + \frac{cut(\mathbf{A},\mathbf{B})}{assoc(\mathbf{B},\mathbf{V})} \tag{2.2}$$

where $assoc(\mathbf{A},\mathbf{V})$ is the sum of the edge weights of set $\mathbf{A}$ to set $\mathbf{V}$. If we reconsider the example in Figure 2.4 the isolated nodes will no longer have small *normalized cut* solutions since the cut will be an appreciable fraction of the total connection of that set to all other nodes.

The criterion has found many research applications and the Normalized Cuts (NC) algorithm was used in one of the first works on over-segmentation [218]. One particular implementation of the NC algorithm, based on the work of Stella Yu, was used was made available by Mori [185] and we use this in our quantitative evaluation.

Unfortunately, minimizing the *normalized cut* criterion exactly produces an NP-complete problem that can only be approximately solved using the method in [238]. In their original paper Shi and Malik [238] propose a spectral approximation to the criterion with complexity $\mathcal{O}(N^{3/2})$, where N is the number of pixels [159]. Their technique only requires the top few eigenvectors of the graph Laplacian and precision can be low which allows them to use the Lanczos method [104] to solve a generalized eigenvalue problem. If the matrices are sparse (sparsity is determined by the size of the spatial neighborhood of nodes in the graph) then runtime can be reduced. In practice they remove 90% of edges from a fully connected graph representing pixels in the image and still achieve good empirical performance. There has also been recent progress on alternative approximation techniques. Sharon et al. [235] introduce a method that achieves a further reduction by a factor of $\sqrt{N}$ and Cour et al. [59] achieve a linear time algorithm using a multi-scale approach.

Despite this success the method has several practical limitations. The computational complexity is prohibitive for very large images, particularly as the number of segments increases [159] and the errors of the different approximations are not currently well understood. For example, the approximation means you are required to map the continuous eigenvalue solution to discrete solutions by employing a heuristic (eg. K-means, see next section.) [258].

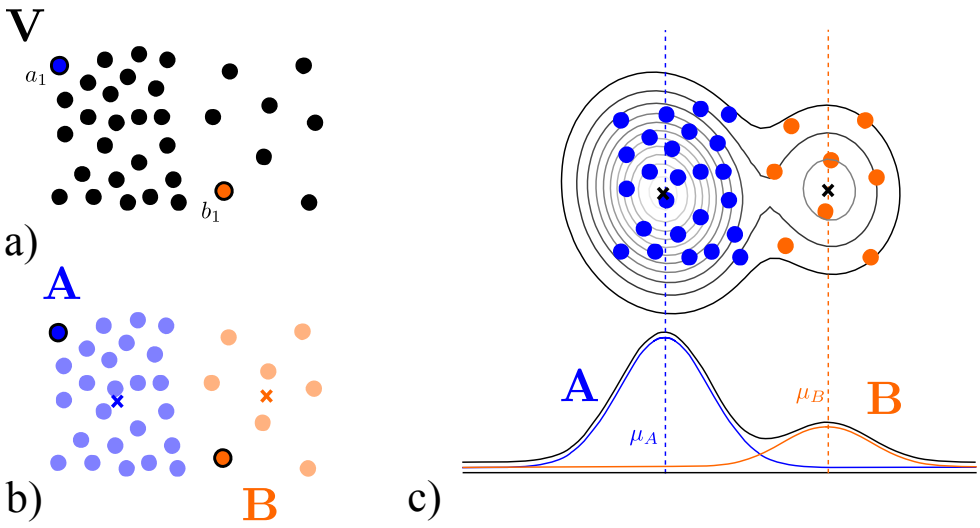


Figure 2.5: Mixture models. a) A set of feature points, $\mathbf{V}$, and an initial assignment of two cluster centers a_1 and b_1. b) Final partition of data using K-means with $K = 2$ with final cluster centers marked with crosses. c) Fitting a gaussian mixture model to data points $\mathbf{V}$ showing cluster means μ_A and μ_B.

2.2.3 Partitional Algorithms

Our last group of clustering methods have a procedure for assigning a set of data points to a fixed set of prototypes/models. By *model* in this context we do not mean a model of an object in a scene but usually a statistical model based on fitting a set of parameters to data.

We begin with the a set of non-probabilistic algorithms the simplest of which is a technique called the K-means algorithm [168]. The algorithm partitions a set of data into some number K of clusters such that the squared error between the empirical mean of a cluster and the points in the cluster is minimized. Unfortunately, minimizing this objective function is known to be an NP-hard problem even in the case when $K = 2$ [76]. The K-means algorithm is therefore a greedy algorithm that converges on a local minimum based on a initial assignment of K clusters. An example of an initial assignment of clusters and a final partition of the data can be seen in Figure 2.5a-b.

The K-means algorithm can be extended in many ways including pre-computing data structures such as projection and metric trees for nearest neighbour search [144]. However, simple K-means is seldom used in practice for image segmentation tasks. Everingham et al. [84] demonstrate that the `FH` algorithm outperforms both graylevel and color versions of the K-means algorithm on several datasets [85].

The requirement to initialize, or seed, the K-means algorithm is used by several other K-way partitional algorithms. One algorithm of particular note is the Random Walker [108]. The algorithm begins with a set of K seeds and then determines the membership of other data points based on how likely a random walk is to arrive at a particular data point from each seed. This can be found by solving the combinatorial Dirichlet problem [110] based on a system of sparse linear equations. Interestingly, this approach has been shown to be equivalent to minimizing the same functional as the *minimum cut*, with the difference that the minimization in the case of the random walker algorithm is performed over the field of real numbers instead of binary values. It can therefore be interpreted as the MAP solution to a Markov random field [241], which we present in the next section. There are also strong theoretical links with the *normalized cut* criterion [109]. This method has several notable advantages over the *minimum cut* criterion: it does not suffer from the "small cut" problem, has provable robustness to noise, extends easily (and exactly) to an arbitrary number of labels and yields a probability that a given node belongs to a segment [107]. The work in this thesis exploits *constraint* edges in minimum cut graph construction (see Chapter 5) and it is currently unclear how to impose the same constraints on the topology of the random walker solution.

Other approaches to seeded image segmentation include region growing methods. Recent approaches include the Turbo pixel algorithm [159] which segments the image by dilating a set of seeds so as to adapt to local image structure until the boundary terminates when two dilating seeds collide. This algorithm has many similarities to early curve evolution techniques [49, 134].

If we consider the feature space representation of an image there is a vast body of work on partitional algorithms in the pattern recognition and machine learning literature [78, 27, 130]. These approaches are based on probability theory which means that the relationships between uncertainties in both model and data can be made explicit. One simple example that extends the K-means approach in this setting is a mixture of K-Gaussians based on assigning data points to two latent variables [27]. The EM algorithm [68] can be used to fit two randomly initialized distributions to data. An example of this is illustrated in Figure 2.5c. Practical examples of applying mixture models to foreground/background segmentation inlcude [246]. However, the `FH` algo-

rithm has been shown to outperform simple Gaussian mixture model (GMM) based on texture features from a bank of 8 Gabor filters [84].

Direct application of many partitional machine learning techniques to image segmentation have been limited because we usually require spatial coherence in the image plane. It is a simple task to turn a feature space segmentation into an image plane segmentation, by assigning spatially disconnected sets of pixels to different segments, but this post-processing is ad hoc.

A more principled approach based on finding coherent sets of labels in the image plane is to use a graphical model of a random field. Random field methods are partitional because they commonly assign each pixel to a finite set of labels $\mathcal{L}$. We make use of this formulation of the segmentation problem in Chapter 5 and review the methodology separately in the next section.

2.3 Labelling

A labelling problem is the task of assigning a particular label from a set $\mathcal{L}$ to every pixel in an image. Labels can be used to represent different semantic classes or discrete approximations to a continuous variable. Applications of labelling problems include estimating: stereo disparity [38], region segments [35, 225], texture elements[147], collage pieces [12] or object classes [138].

The principal representation of labelling problems in vision research is a graphical model of a random field. In a random field model, nodes in the graph represent random variables that can take on values from a discrete label set. Typically, the random variables correspond to pixels in an image and unlike previous sections we will assume that methods presented here apply to a segmentation of the image plane. The graphical model provides a neighbourhood relationship between variables that incorporates not only uncertain image measurements, but also a prior model over the labelling space in a principled manner.

Two important classes of undirected graphical models used in the literature are Markov random fields (MRF) and conditional random fields (CRF).

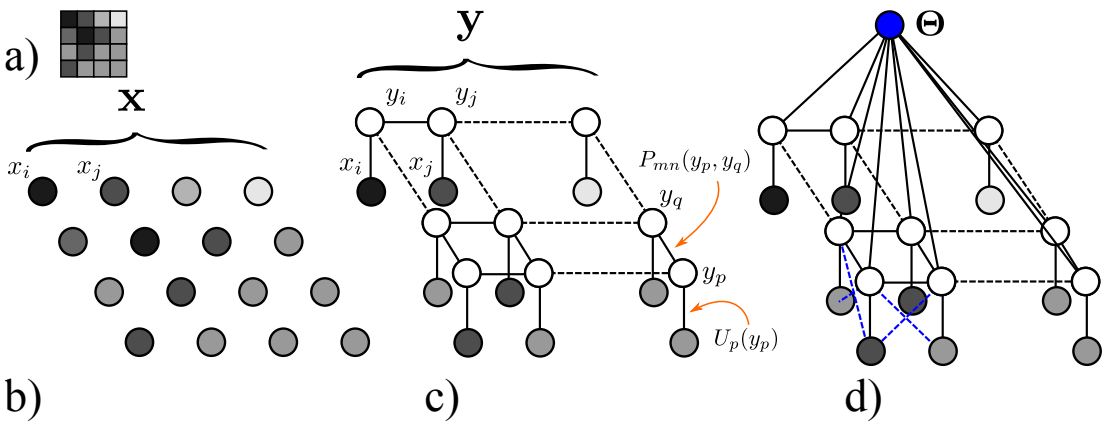


Figure 2.6: Random fields models. Graphical model representations of Markov and conditional random fields. a) Example 4×4 pixel image. b) Nodes representing a set of pixel data $\mathbf{x}$. c) A field of image labels $\mathbf{y}$ at each site from the set $\mathcal{L}$. d) This simple model can be augmented using additional terms, eg. contrast terms [145], or more generally with priors $\boldsymbol{\Theta}$ on the labels, eg. [283].

2.3.1 Random Field Models

Random Field models originated in the physics of spin-glasses and were introduced for use in image modeling by Geman and Geman [101]. We will make use of a Markov random field (MRF) formulation in Chapter 5 and introduce the basic notation used in the thesis here. An MRF defines a generative model of the joint probability distribution over observed data, $\mathbf{x}$, and a set of hidden random variables $\mathbf{y}$, denoted Pr($\mathbf{y}$,$\mathbf{x}$). From Bayes' rule, the joint probability is equal to the product of the likelihood and prior probabilities:

$$\Pr(\mathbf{y}, \mathbf{x}) = \Pr(\mathbf{x}|\mathbf{y})\Pr(\mathbf{y}) \tag{2.3}$$

where Pr($\mathbf{x}|\mathbf{y}$) is the likelihood and Pr($\mathbf{y}$) is the prior. Let us consider a discrete random field Y defined over sites $\mathcal{S} = \{1, 2, \ldots, N\}$ with neighbourhood system $\mathcal{N}$. Each random variable can take a value from a finite label set $\mathcal{L} = \{l_1, l_2, \ldots, l_k\}$. A random field of the joint distribution is *Markovian* if the probability of an assignment of a random variable depends only on the neighbouring random variables given by $\mathcal{N}$. From the Hammersley Clifford theorem [113], the posterior distribution $Pr(\mathbf{y}|\mathbf{x})$ is a *Gibbs* distribution [153]:

$$\Pr(\mathbf{y}) = \frac{1}{Z}\exp\left(-\sum_{c\in\mathcal{C}}\psi_c(\mathbf{y_c})\right) \tag{2.4}$$

where $\psi_c(\mathbf{y_c})$ are potential functions defined over variables constituting the clique c, Z is a normalizing constant known as the *partition function*[1] and $\mathcal{C}$ is the set of all cliques.

A collection of all pixel/label assignments $\mathbf{y}$, called a *configuration* or a *labelling*, takes values from the set $\mathbf{L} = \mathcal{L}^N$. The corresponding *Gibbs* energy associated with a configuration is given by:

$$E(\mathbf{y}) = -\log \Pr(\mathbf{y}|\mathbf{x}) - \log Z = \sum_{c\in\mathcal{C}}\psi_c(\mathbf{y}_c) \tag{2.5}$$

The maximum *a posteriori* (MAP) configuration $\mathbf{y}^*$ of the random field is defined as:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}\in\mathbf{L}} \Pr(\mathbf{y}|\mathbf{x}) = \arg\min_{\mathbf{y}\in\mathbf{L}} E(\mathbf{y}) \tag{2.6}$$

Let us consider the common case of a first order MRF with a site for each pixel from a set $\mathcal{P}$ of an image with two potentials in the energy function:

$$E(\mathbf{y}) = \sum_{p\in\mathcal{P}} U_p(y_p) + \sum_{p,q\in\mathcal{N}} P_{pq}(y_p, y_q) \tag{2.7}$$

The data terms U_p represent the cost for pixel p having label y_p and corresponds to the negative log likelihood of a label being assigned to pixel. Unlike the unary terms which depend on data the pairwise potential terms P_{pq}, defined on a neighborhood system $\mathcal{N}$, encourage similarity between labels. For example, variations of the Potts model [138] are common which gives a low energy when $y_p = y_q$ and penalizes with a high energy otherwise. An illustration of the model just described can be seen in Figure 2.6.

The problem of finding the MAP solution of a general MRF problem is NP-hard [38] but there exist several algorithms which can compute exact [35, 141] or strong local minima [38] in polynomial time. In particular, the term "Graph Cuts" has come to refer to the application of st-MINCUT algorithms to find the MAP solution to an MRF. This method was first proposed by Greig et al. [111].

A conditional random field (CRF) is a discriminative field that uses the same framework as an MRF but where the potential functions are conditioned on the data

[1]The letter Z stands for the German word *Zustandssumme* meaning "sum over states".

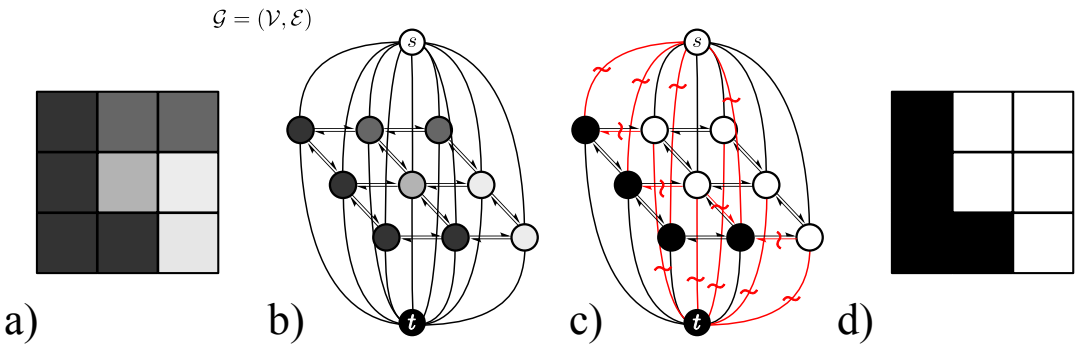


Figure 2.7: Graph Cuts. a) 3×3 pixel image. b) Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that represents image [a]. c) Example *minimum cut* on graph that separates source, **s**, and sink, **t**. c) Binary solution that the *cut* in [c] represents.

[151]. When these functions on the observed data are fixed, the rest of the model on the hidden variables is indistinguishable from an MRF model [129]. Therefore, it is common in the literature to refer to a CRF model as one where the observed variable depends on more than one pixel, even if at the level of inference the machinery is identical to that presented for MRFs. In the following we present methods for setting potential functions based on distributed sets of pixels.

2.3.2 Potential functions

In Equation 2.9 there are two terms in the MRF/CRF energy. The first term, called the *unary* term U_p, represents the cost of a pixel belonging to a particular label. Methods for calculating this type of potential that have proved very successful for recognition tasks include *TextonBoost* by Shotton et al. [239]. Rather than use an explicit model for each object in the scene (eg. [143]) this algorithm learns a joint model for texture and shape based on a boosted combination of texton features inside windows of variable position and size.

The second term, the pair wise term P_{pq}, represents the cost of two neighboring pixels taking similar or different labels. For example, it is common for pairwise potentials of a CRF to take the form of a Potts model:

$$P_{pq} = \begin{cases} 0 & \text{if } x_p = x_q \\ g(p,q) & \text{otherwise} \end{cases} \tag{2.8}$$

where the function $g(p,q)$ estimates the affinity between adjacent pixels p and q. In a

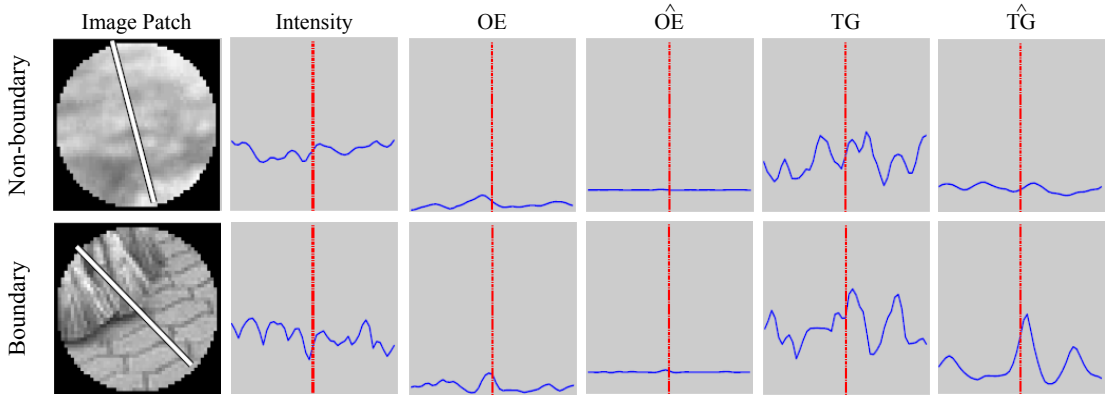


Figure 2.8: Pb feature responses, reproduced from Martin et al. [174]. In each row, the first panel shows the image patch. Details of each feature can be found in the paper. The challenge posed in creating the Pb detector is to combine the response from different cues to detect and localize boundaries. See [174] for more details.

contrast sensitive Potts model the function $g(i, j)$ is commonly of the form:

$$g(p, q) = \theta_p + \theta_v exp(-\theta_\beta ||I_p \cdot I_q||^2) \tag{2.9}$$

where the I_p and I_q are the colour vectors of pixel p and q respectively and parameters θ are learned using training data [35, 225, 239].

However, there is a recent body of work that uses a region of support around the surrounding pixels to produce an estimate of the presence of a *natural* [174, 74] or *occlusion* [124] boundary. Figure 2.8 illustrates the difficulty of deciding the presence of a boundary in a small image patch. We observe little evidence for a boundary in the first four cues (details can be found in [174]) but a greater change in the cue $\hat{TG}$. This example illustrates the need to combine different cues and learn from data and we briefly discuss some of the algorithms that can be used to estimate this pairwise association from data that are used in subsequent chapters of the thesis.

Early approaches to this problem usually focused on "edge" detection. The Canny edge detector [43] is a widely used algorithm that models boundaries in an image as brightness step edges. While very successful in many applications the Canny criteria for an edge causes problems in natural images due to the ubiquitous phenomenon of texture. It therefore tends to produce a strong response in regions of an image that a human would not associate with a boundary or fails to produce a response to an edge

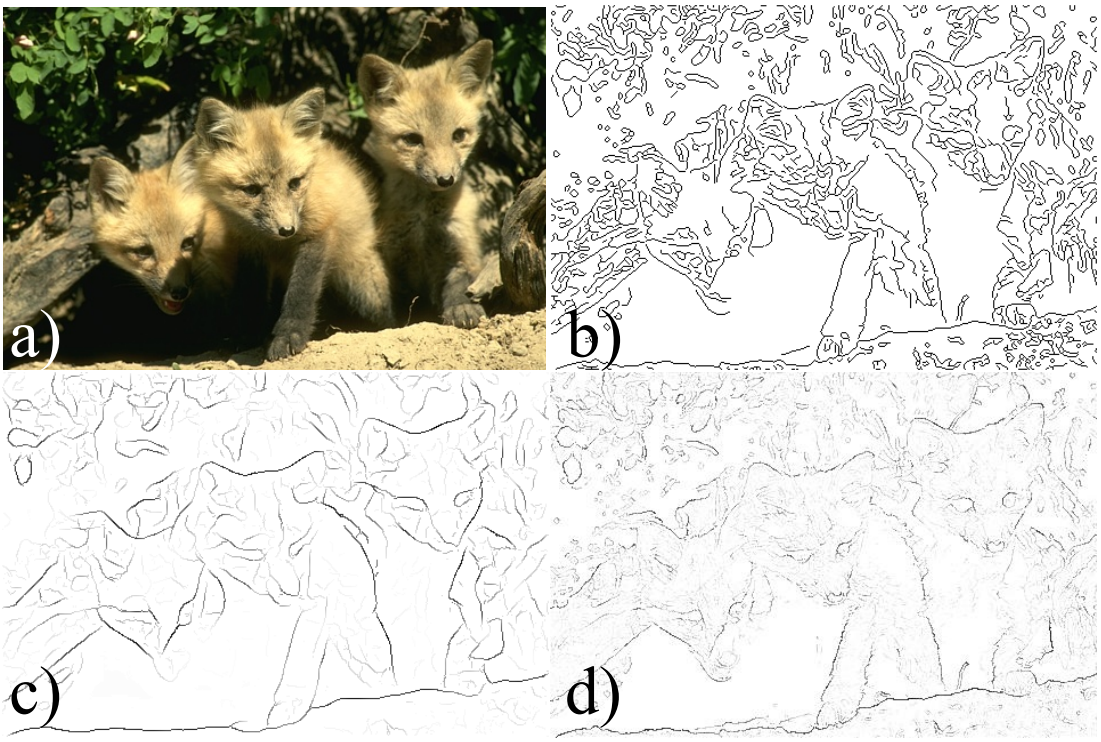


Figure 2.9: Example boundary detectors. a) Original image. b) Canny edge detector [43] c) Pb boundary detector [174] d) BEL boundary detector [74].

in a region with very subtle changes in brightness. An example image of the boundary map for the Canny detector can be seen in Figure 2.9b.

The term *natural* boundary was coined in the work of Martin et al. [174] to refer to the boundaries that are normally selected by humans from natural images. Their `Pb` detector combines the response from several different hand selected cues in a supervised learning framework. Features include oriented energy and texture gradients. This algorithm is used in the first application of superpixels for weighting edges in a graph partitioned using the `NC` algorithm [218]. An example image of the boundary map for the `Pb` detector can be seen in Figure 2.9c.

Another recent algorithm is the "Boosted Edge Learning" (`BEL`) algorithm of Dollr et al. [74]. The `BEL` detector tries to incorporate low, mid and high level images cues in a single detector in a supervised learning paradigm. Unlike the `Pb` detector it does not use designed local features (texture gradients, brightness gradients...) but focuses on learning the required features from labeled training data. The features include gradients at multiple scales, differences between histograms computed over filter

responses and Haar wavelets all computed using integral images [270]. The number of training examples based on this feature set is high ($\mathcal{O}(10^8)$ samples).

This data set is then used to train a "probabilistic boosting tree" (PBT) [261] which is similar to a decision tree with boosted weak classifiers at each node. By using a boosting framework it is possible to select different features that are discriminative at different levels in the tree. Given a trained tree and a new test example an estimate of the posterior is computed recursively down the tree until we reach a leaf node. The output is simply the learned empirical distribution at the node.

The authors demonstrate the versatility of the method by learning toy models of parallelism and modal completion, as well as demonstrating good performance on the BSD dataset. We see further evidence of good performance in Section 2.5.3 after we introduce different datasets in Section 2.5.1. Alternative approaches to learning edges include [208] where they exploit object models to help guide the final segmentation but we shall see in the next chapter that it is difficult to exploit this approach for our research problem.

2.4 Applications of Superpixels

We now consider specific examples of the application of segmentation techniques in the literature with a particular focus on methods that exploit a superpixel representation. A summary of pipeline applications of superpixels grouped by the underlying algorithm is given in Table 2.1. Methods are presented in approximately chronological order.

Early work on using small segments to aid image interpretation includes [269, 192, 251]. Vincent and Soille [269] use the watershed algorithm for image segmentation on medical and digital elevation images although their original aim is a final correct segmentation and they consider the resulting "over-segmentation" to be a disappointing result. Najman and Schmitt [192] extend the watershed approach to hierarchical segmentation. This allows them to chose the level of granularity in a principled way and therefore control the over-segmentation. They apply the method to indoor and outdoor images and also exploit the properties of segments to evaluate the presence of an object/background boundary. Tabb and Ahuja [251] also introduce a multi-scale image segmentation scheme, similar to watersheds, based on the structural relationship between regions rather than more common scale-space methods [164]. The authors apply

it to natural images, aerial scenes and 2-D slice CT scans. Whilst the method can give certain structural guarantees based on a dendrogram of regions they are different from the structured segmentation that we present in the thesis.

A multiscale approach is also adopted by Sharon et al. [235] based on a solution to the *normalized cut* criterion. Their approach reduces the complexity of the original method [238] and they demonstrate useful performance on real world images where segments sometimes corresponds to semantic regions. However, the nodes corresponding to image segments do not have to lie on a regular grid, giving rise to an irregular pyramid. In a subsequent paper [236] the authors exploit their previous approach to combine multiscale measurements of intensity contrast, texture differences and boundary integrity to improve the segmentation of regions separated by weak but consistent edges. Their approach demonstrates image cues integrated over regions improve the segmentation process and they qualitatively compare regular (10×10) sampling and their irregular lattice to show that their approach captures better object boundaries. Unfortunately, they sacrifice the useful properties of the regular lattice.

An approach that exploits superpixel regions for stereo computation is presented in [252]. They use superpixels to produce a compact depth representation that makes their framework computationally feasible. They use an early version of the `MS` algorithm and incorporate superpixels in a global matching framework that includes scene constrains for regularizing the solution. The approach particularly helps depth estimation in textureless regions. However, as the relationship between superpixel is unknown they must use a greedy search strategy when comparing depth to increase the speed of their approach. Recent examples of using superpixels in stereo using the `FH` algorithm include [180].

Ren and Malik [218] coined the term "superpixel" in a paper that showed that it was possible to learn a classification model for image segments based on Gestalt cues. They also put forward the argument for considering over-segmentation as a preprocessing step and their motivations are discussed in greater detail in Chapter 3. Results in the paper demonstrate that curvilinear continuity between segments is an important cue for distinguishing a "good" segmentation.

The effect of over-segmentation and feature choice is also investigated by Barnard et al. [22]. The authors present a novel system for attributing a set of annotating words

to image segments for the purpose of automatic description of images. They attempt to aggregate over-segmented regions based on the posterior probability of segments belonging to similar words. They evaluate a Gaussian mixture model (Blobworld [48]), `MS` and `NC` algorithms and demonstrate that their system produces useful results on several datasets. However, although they demonstrate the ability to successfully merge segments based on word prediction, their results show that a better initial segmentation improves overall performance and their approach does not offer a method for doing this.

The *Lazy Snapping* approach of Li et al. [163] uses a superpixel representation in an interactive image cut-out tool to coarse and fine scale processing. It uses a *Graph Cut* formulation on a pre-computed over-segmentation using the watershed algorithm [269] to increase efficiency and make interaction rates near real time. Later papers that revisit work on interactive video cut-out using superpixels include [272, 266].

In their original paper on using the minimum spanning tree for segmentation Felzenszwalb and Huttenlocher [89] describe a large-scale image database application that computes image features based on superpixel regions [217]. However, their approach uses a 'bag-of-features' so they do not exploit the spatial relationship between superpixels to classify an image.

A more principled approach is developed by Borenstein et al. [33] who combine top-down and bottom-up methods based on earlier work [236]. They use an over-segmentation of the image along with prior knowledge about an object in the form of shape fragments learned from training data. The main advantage of their approach is the ability to use top-down information to group together segments belonging to the object despite image-based dissimilarity and conversely break apart homogenous segments that contain both figure and background regions. This approach produces good results but it produces an irregular segmentation pyramid. This means that only the relationship between superpixel within the hierarchy can be used rather than the relationship between superpixels in the image plane. Additionally, very specific class information is not always available. We investigate an alternative weaker prior model in Chapter 4. The authors also explore an approach to learning the figure-ground fragments in a semi-supervised framework [32] which depends on a superpixel segmentation to serve as an initial approximation to object parts.

	Algorithm	Pipeline Application
Superpixel segmentations	**Mean Shift**	Tao et al. [252]
	Comaniciu and Meer [53]	Wang et al. [272]
		Yang et al. [287]
	Normalized Cut	Ren and Malik [218] Barnard et al. [22]
	Shi and Malik [238]	Mori [185] He et al. [116]
		Yang et al. [286] Cour and Shi [58]
		Rabinovich et al. [213]
		Rabinovich et al. [214]
		van den Hengel et al. [266]
	Sharon et al. [235]	Sharon et al. [236]
	Spanning Tree	Batra et al. [25]
	Felzenszwalb and Huttenlocher [88]	Gould et al. [105]
		Saxena et al. [230]
	Watershed	Li et al. [163]
	Vincent and Soille [269]	Najman and Schmitt [192]
		Arbelaez et al. [19]
		Micusik and Kosecka [179]
	Others	Zhu et al. [295]
	Tabb and Ahuja [251]	Todorovic and Ahuja [257]
		Ahuja and Todorovic [15]
	Levinshtein et al. [159]	
	Deng and Manjunath [70]	
	Soille [242]	
	Moore et al. [182]	Choi et al. [51][275]
	Sargin et al. [229]	
	Vedaldi and Soatto [267]	Fulkerson et al. [97]
	Rabinovich et al. [212]	Galleguillos et al. [99]
		Galleguillos et al. [100]
	Veksler et al. [268]	
Multiple segmentations	**Mean Shift**	Kohli et al. [138]
	Comaniciu and Meer [53]	Ladicky et al. [148]
	Spanning Tree	Hoiem et al. [121] Hoiem et al. [118]
	Felzenszwalb and Huttenlocher [88]	Hoiem et al. [119] Russell et al. [227]
		Hoiem et al. [124] Hoiem et al. [120]
	Combinations	Malisiewicz and Efros [172]
		Pantofaru et al. [200]
		Gould et al. [105]

Table 2.1: Example superpixel algorithms and recent applications exploiting them in the vision processing pipeline. Applications include classification, labelling, interactive image editing and stereo.

Mori [185] also investigates using a part-based model using superpixels. The author investigates human pose models and uses the superpixels to constrain the pose estimation search. The superpixel centers are used to restrict the joint positions in the model and the superpixel support is used to compute image features. They generate superpixels using the NC algorithm. The approach achieves good results for half-limb pose estimation and can use the superpixels to produce partial segmentations. However, the superpixels are fixed in their approach and there is no mechanism for altering superpixels based on updated estimates of the pose model.

One of the first papers to explore the limitations of a fixed superpixel segmentation was the work of Hoiem et al. [121]. In this paper the authors exploit superpixel regions of support to create feature vectors for estimating geometric cues from a single image. A similar pipeline to that exploited in this paper was illustrated in Figure 1.3. They claim that estimating the orientation of large-scale surfaces requires complex geometric features that must be evaluated over fairly large regions in the image and use multiple superpixel segmentations based on the FH algorithm [89]. Their image cues consist of 24 separate features based on texture, shape, position and lines of intersection of superpixels. In further papers Hoiem et al. [118, 119] extend the use of multiple superpixel segmentations to include *constellations* – groups of superpixels. For each constellation, they estimate the likelihood of each of three possible labels and the confidence that all of the superpixels in the constellation have the same label. Each superpixel's label is then inferred from the likelihoods of the constellations that contain that superpixel. Their method produces impressive results and motivated the work on superpixels in this thesis. However, the authors use a heuristic search in the parameter space of the FH algorithm to produce multiple hypotheses. Although we do not exploit uncertainty in superpixel solution or multiple segmentations in this thesis, the framework we present in Chapter 5 allows these questions to be addressed in a principled manner.

He et al. [116] use superpixels as the basis for a CRF formulation that imposes category-based information on the label field. They use the NC algorithm to oversegment an image and use the superpixels to create feature vectors which are subsequently merged by assigning labels that correspond to object categories. However, in an earlier paper [115] they make use of the spatial relationship between label regions in

a pixelwise CRF. They are unable to achieve this when using superpixels because the spatial relationship varies from image to image, which originally motivated the work in this thesis. This is discussed in more detail in Chapter 3. The superpixel support in their approach is again static and the segmentation is only alterable by merging.

One approach that tries to learn the underlying over-segmentation based on unlabeled image sets is [257]. They make use of an earlier approach of Tabb and Ahuja [251] that produces a segmentation tree and uses it for the unsupervised identification of photometric, geometric, and topological (mutual containment) properties of multi-scale regions defining objects in the category. They produce useful results on several databases but are restricted because their tree-based representation can only model mutual containment rather than more complicated topological relations that are found in object classes. Russell et al. [227] also introduces an approach that tries to learn visually similar object and scene classes together with their image segmentations based on a large collection of over-segmented images. They set up their task as sifting through a "soup" of segments to separate good segments from bad, for each discovered object category. They make use of the `NC` algorithm and `Pb` boundary detector [174] and their method discovers several object classes from the Caltech, MSRC and LabelMe databases. However, there is no clear way to improve the underlying segmentations based on their learned models.

The idea of "segment soup" is further explored in the work of [172]. They set out to answer two questions 1) How important is good spatial support for recognition? and 2) Can segmentation provide better spatial support for objects? They answer the first question by comparing recognition performance using ground-truth segmentation vs. bounding boxes and demonstrate that segmentations improve performance for several classification algorithms. To answer the second question, the authors use the multiple segmentation approach to evaluate how closely real segments can approach the ground-truth for real objects. They compare `NC(Pb)`, `FH` and `MS` algorithms. They conclude that finding the right spatial support for objects improves performance and that different segmentation strategies are beneficial for different object types. Additionally, the idea of using the superpixel region of support to bin keypoints [170] for object class recognition is explored in [287]. The benefit of segment *stability* using multiple segmentations is demonstrated in [212, 213, 99].

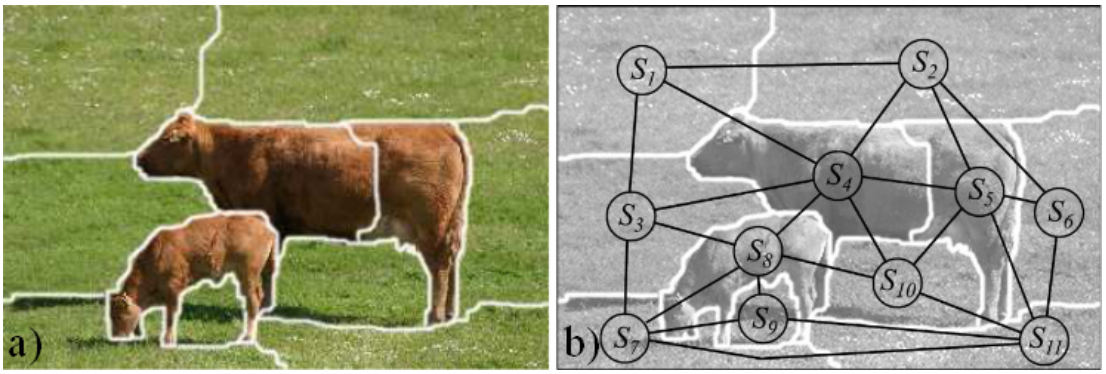


Figure 2.10: Superpixels and CRFs. a) A partition of the image into superpixels based on appearance characteristics using the MS algorithm. b) CRF structure over superpixels. Reproduced from [105].

Other novel cues that a superpixel representation has enables includes occlusion boundaries [230, 124, 229], class-specific affinities [25] and segment consistency [138, 200]. The relative location of superpixel labels is exploited in [105]. The authors try to model the spatial relationship between object labels. For example, identifying "cow" pixels indicates that pixels above and to the sides are more likely to be grass rather than say "car", see Figure 2.10. However, the spatial relationship between nodes representing superpixels varies, see Figure 2.10b, and the authors resort to superimposing a regular grid on the original image for computing *relative location* features. Co-occurrence of superpixel labels is also shown to be a useful cue in [100, 97, 180]. Ladicky et al. [148] present an associative hierarchical CRF framework that incorporates multiple superpixel segmentations in a principled manner.

Recently there has been renewed interest in the underlying algorithms used to generate superpixels based on low-level and mid-level cues. Arbelaez et al. [19] present a method for merging watershed segments based on a boundary estimate and an Ultrametric Contour Map [18]. The method produces a hierarchy of segments and uses an estimate of straight line continuity. The authors demonstrate good performance on the Berkeley Segmentation Dataset (see next section) using several evaluation measures. However, the method does not restrict the topology of segmentations and it is difficult to give a proper probabilistic interpretation.

One approach to superpixels that uses an MRF framework similar to the approach in Chapter 5 is the work of Veksler et al. [268]. They present two methods to produce

either compact or constant intensity superpixels and apply their method to salient object segmentation problem using the binary segmentation algorithm of [34]. However, their approach only makes limited use of the unary potentials and unlike our method they do not place any restrictions on the topology of the superpixel lattice. They also investigate the temporal aspects of the model by segmenting video footage to produce "supervoxels" in a similar manner to our approach in Section 3.10.

Finally, the utility of superpixels is also illustrated in a recent application of linear clustering approaches to superpixels for object class recognition and medical image segmentation [11]. The authors learn a CRF model based on the work of [105] and demonstrate improved pixel wise error. They also apply their technique to neural electron microscopy images and use a feature vector based on SIFT descriptors [170] and an SVM classifier to label superpixels cell or non-cell. They include this in a graph cut based framework to produce a final segmentation on large images (2000×1500) and conclude that the superpixel segmentation improves performance.

2.5 Datasets and Evaluation

With the variety of different algorithms and approaches at our disposal it is often necessary to be able to demonstrate the advantages of one over the other. Often it is possible to make analytic comparisons - for instance computational complexity - but commonly we also require an empirical evaluation of performance. In Section 2.5.1 we present several standard datasets that we use for the quantitative evaluation of the algorithms presented in the thesis and in Section 2.5.2 we introduce our evaluation methodology. Further details are given in Appendix C. Finally, in Section 2.5.3 we highlight the different performance of the boundary detection algorithms across datasets.

2.5.1 Datasets

The datasets we have used for the evaluation of algorithms presented in the thesis are all challenging real-world datasets. They have also been chosen to contain differing modes of variation. For example, two datasets contain unconstrained viewpoints whereas the third is captured from the viewpoint of a front seat passenger of a car. We discuss each in turn.

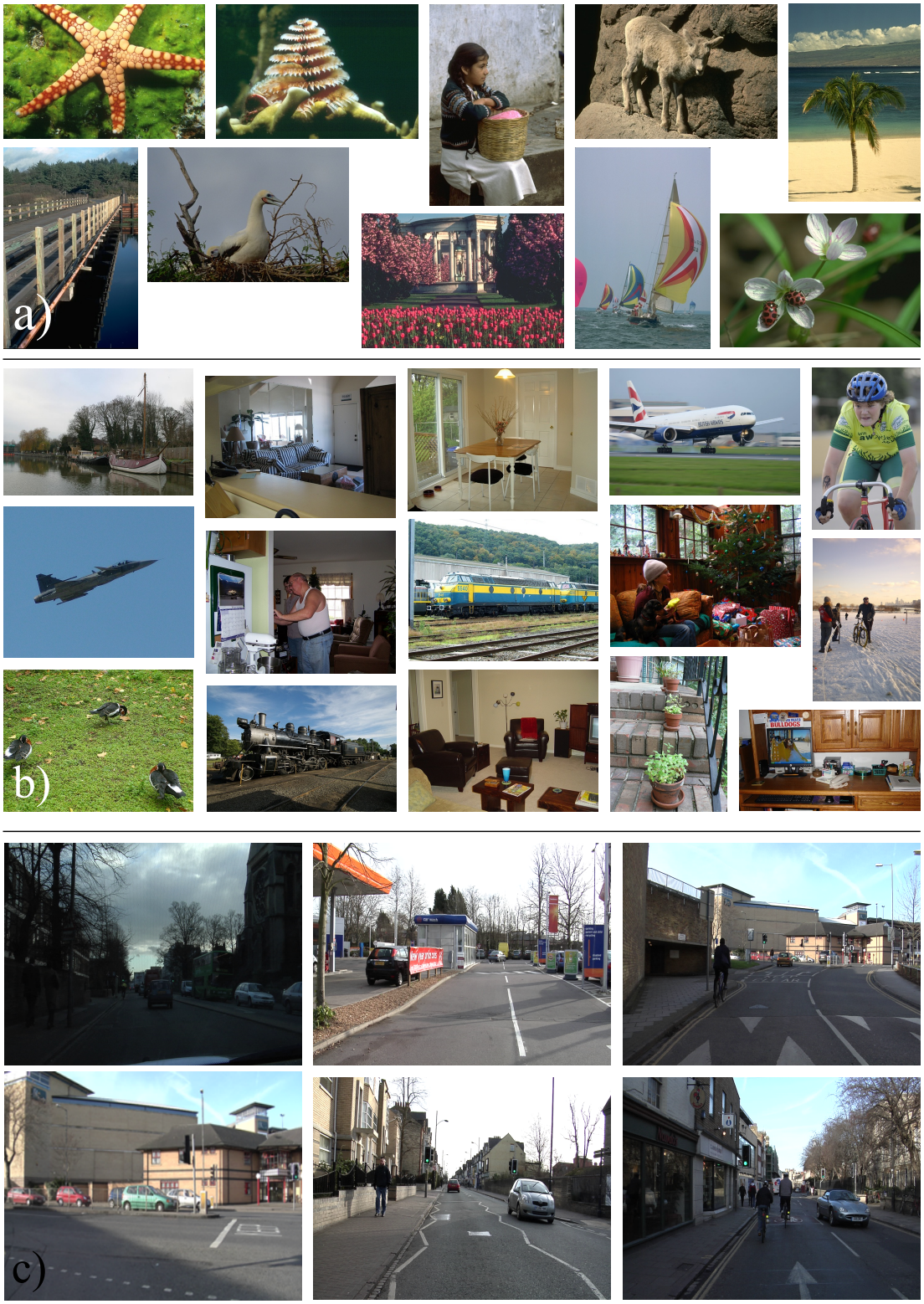


Figure 2.11: Example images from different datasets. a) Berkeley Segmentation Database [174] b) PASCAL Visual Object Classes dataset [87] c) CamVid dataset [39].

Berkeley Segmentation Dataset

The Berkeley Segmentation Dataset (BSD) public benchmark consists of images of a wide variety of natural scenes, both grayscale and color, with segmentations for 300 images provided by a possible 30 human subjects [173]. These subjects were instructed to:

> *Divide each image into pieces, where each piece represents a distinguished thing in the image. It is important that all of the pieces have approximately equal importance.*

This has resulted in images with a mean number of ~ 20 separate regions in the ground truth and an average of ~ 5 ground truth segmentations per image. It was intended to demonstrate the different perceptual organizations of scenes for different human subjects and additionally allows the investigation of the degree of consistency between segmentations of the same region. The images are divided into a training set of 200 images, and a test set of 100 images. Example images can be seen in Figure 2.11a.

The scenes in the BSD consist of a wide variety of different settings including landscapes, human activities and wildlife amongst others. The capture conditions are also unconstrained, from wide-angle landscapes to close-up crops on parts of objects and occasional ariel images. There are no controlled lighting conditions in the dataset. This entails that the set of objects is largely unconstrained and there are no class labels in the ground truth data. This makes its use in recognition tasks limited as it is difficult to identify object similarities between images, although a limited subset has been used for evaluating figure ground separation [220]. When selecting the benchmark images with "one discernable object" were used. This was intended to remove difficult examples, eg. photographs of reflections or close up examples of texture, but there are also noticeably fewer complex scenes, ie. crowds or cluttered urban environments, in this dataset.

The BSD has become a standard dataset for evaluation of segmentation and boundary detection algorithms, as evidenced by its widespread use [219, 90, 74, 18, 294, 221, 171]. The publishers of this dataset also present benchmark results for submitted algorithms and keep and ranked list of images based on performance. In preliminary experiments in Chapter 3 we make use of a subset of eleven images of the test dataset,

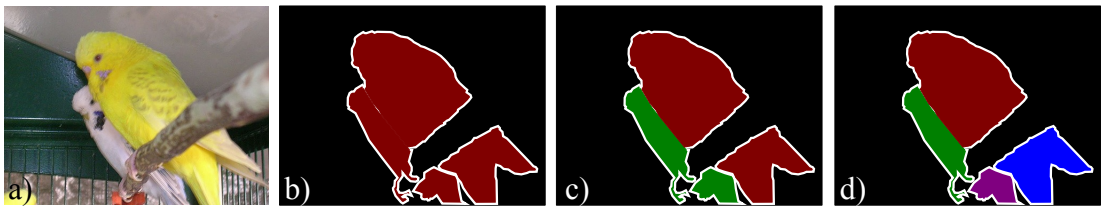


Figure 2.12: Example images from PASCAL Visual Object Classes dataset showing ground truth labels. a) Original Image. b) Class labels c) Object instance labels d) Instance region labels (generated from c). Mask label is shown in white and unlabeled regions in black. Unlike BSD data only selected objects (regions) are labeled and therefore $\sim 76\%$ of pixel are unlabeled.

spaced uniformly on this ranked list - which we refer to as BSD11. We use the color segmentations in our evaluation.

PASCAL Visual Object Challenge

The PASCAL Visual Object Classes (VOC) challenge is a benchmark in visual object category recognition and detection for the vision and machine learning communities [87]. Every object from one of 20 classes are labeled in every image in which they appear. The 20 classes are taken from a taxonomy with four main branches - vehicles, animals, household objects and people. Example images can be seen in Figure 2.11b. The images were collected from the flickr[2] photo-sharing website which results in a "unbiased" dataset - ie. unlike the BSD they were not manually selected for a specific task.

Objects in each image are labeled from a set of 20 classes and each image contains a mean number of ~ 4 separate object instances. Development of the database is ongoing and new challenges are run each year. The work presented in this thesis uses the train and validation sets of the segmentation task 2007-2008, composed of 1023 images. This subset has been used in recent work on evaluating segmentation performance [19]. Note that because of the restriction to only 20 classes within each image $\sim 76\%$ of all pixels are unlabeled. There is also a *mask* used to highlight difficult or ambiguous pixels at the boundary of objects. We ignore these difficult pixels from our analysis.

The Pascal VOC dataset is one of the most difficult and varied for recognition. The scene variability is partially constrained by the sets of possible objects and the set

[2]http://www.flickr.com/

of objects that are labeled in the dataset are restricted to those 20 of interest. However, the dataset contains significant variability in terms of object size, orientation, pose, illumination, position and occlusion. The capture conditions are are unconstrained. We evaluate performance using the object instance labels provided. An example comparing class, object instance and region ground truth images can be seen in Figure 2.12.

Cambridge-driving Labeled Video Database

The Cambridge-driving Labeled Video Database (CamVid) [39] is a collection of high definition videos with object class labels from 32 classes - captured from the perspective of a passenger in a driving car. The road driving scenes produce cluttered complex urban scenes and includes ego-motion of the camera. The still images taken from the video contain a mean number of ~ 60 separate regions in the ground truth data. However there are no instance labels in this dataset so object instances consist of just those class instances that are separable in a binary mask of the class label. This high number of regions in each image gives one indication the difficulty of this dataset.

The ground truth for this dataset is labeled with "paint tool" that can sometimes lead to lower quality boundaries in some regions of the image. To generate the estimate of ~ 60 separate regions we used morphological closing to remove spurious regions of each binary label mask that makes up the ground truth image using a disc shaped structing element 7 pixels wide. While this is likely to occasionally merge separate regions it significantly reduces the number of spurious regions that are present in the ground truth. We feel it is a reasonable estimate - and serves to demonstrate the added complexity of this dataset.

The capture conditions from a moving car constrain the scene geometry and the distribution of objects. We shall see in Chapter 4 that we can exploit uniformity of constrained capture conditions of this dataset to improve segmentation performance. Of particular note is the inclusion of two illumination conditions - dusk and day - although we do not exploit this in work presented in this thesis. The ground truth consists of 701 stills and is divided into four video sequences (in minutes:seconds): 0001TP (8:16), 0006R0 (3:59), 0016E5 (6:19), and Seq05VD (3:40). In the work in this thesis we follow [40] and use sequences 06R0 and 16E5 for training and 05VD for testing. Examples of the CamVid dataset can be seen in Figure 2.11c.

2.5.2 Evaluation Methodology

In Section 2.4 we reviewed many different pipeline applications of superpixels. One possible evaluation methodology is to gauge the successes of the segmentation component based on the performance of the larger computer vision system (e.g., tracking, recognition, etc.). However, this strategy for comparison of algorithms can become unfair and possibly inconsistent when evaluating systems that are tailored to different applications [264]. Therefore, we choose to evaluate segmentation in its own right and pursue an evaluation methodology based on comparing competing methods directly.

Zhang [293] provides a survey of evaluation methods, and divides them into three categories: *analytical*, *empirical goodness* and *empirical discrepancy*. *Analytical* methods are useful for evaluating runtime or the theoretical behaviour of an algorithm independent of data, eg. [88]. This analysis has been successful in the domain of edge detection [43] but the lack of a general theory of segmentation has meant that analytical methods have proved less useful in predicting behaviour in segmentation tasks [84]. *Empirical goodness* methods make an evaluation of performance without *a priori* knowledge of a particular desired segmentation result. In Chapter 3 we introduce a measure based on greylevel uniformity that we refer to as *explained variation*. However, in general these measures are at best heuristic, and can exhibit a strong bias towards one particular algorithm, see Section 3.5.1.

We therefore exploit an evaluation strategy based on *empirical discrepancy*. In the thesis we adopt the evaluation framework of Everingham et al. [84] by combining several performance measures based on human labeled ground truth data. Everingham et al. [84] note that it is common for different algorithms to perform better or worse under different measures and that it is often difficult to see the trade off between different measures considered in isolation. The authors therefore propose to use a combination of different measures in a fitness/cost function:

$$H(a_{\vec{p}}, \mathcal{D}) = \Phi\left(f_1(a_{\vec{p}}, \mathcal{D}), \ldots, f_m(a_{\vec{p}}, \mathcal{D}), c_1(a_{\vec{p}}, \mathcal{D}), \ldots, c_n(a_{\vec{p}}, \mathcal{D})\right) \tag{2.10}$$

where $a_{\vec{p}}$ is a segmentation algorithm a with parameters $\vec{p}$ and $\mathcal{D}$ is a set of ground truth images. The functions $f_i(a_{\vec{p}}, \mathcal{D})$ are individual fitness functions defined to increase monotonically with some measure of the algorithms' behaviour. Functions $c_i(a_{\vec{p}}, \mathcal{D})$

similarly increase with the cost of some measure of the algorithm's performance. These could be equivalently defined as negative fitness functions but the distinction is useful for reasons of clarity.

The term Φ combines these separate fitness and cost functions into an overall measure of performance. One common example of Φ encountered in detection theory is the combination of sensitivity and specificity used in ROC analysis. In this case the sensitivity and specificity define a point in the the 2D fitness/cost space and a linear weighting of the two is justifiable [195]. A second example might be setting $f_1(a_{\vec{p}}, \mathcal{D})$ to a measure of the accuracy of a segmentation and $c_1(a_{\vec{p}}, \mathcal{D})$ to the running time of the algorithm, with Φ being some weighted sum of the two.

The trade off between different measures of performance is still an ongoing area of study. A useful comparisons of competing evaluation measures can be found in Sokolova et al. [244] and problems of evaluating algorithms over multiple datasets is examined in Demsar [69]. However, there are several performance measures that are regularly used for comparing the performance of segmentations algorithms by comparing one segmentation $\mathcal{S}$ to another $\mathcal{S}'$ where one of these segmentations is considered ground truth. Freixenet et al. [96], and several other authors [19, 188], divide these measures into two types: *region* measures, *boundary* measures.

Region measures compare two segmentations based on the properties of all the pixels within separate segments. In the thesis we will adopt five region measures to compare and contrast the performance of competing algorithms that are commonly found in the literature [19, 56]: Cover Score [19], Rand Index [216], Global Consistency Error [173], Variation of Information [178] and Accuracy [182]. Additionally, we use one of our own measures of performance [183]. This measure, F_{sd}, is used to demonstrate the trade-off between the accuracy of segmentation and detection. For example, we can have good segmentation performance (most pixels in superpixel regions belong to the dominant class) and poor detection (several instances of a particular class are missed). This is illustrated in Figure 2.13b. Further details for the performance measures, including the one we introduce, can be found in Appendix C.

Boundary measures compare the outline of segmented regions to ground truth regions. One particular method for measuring boundary detection was proposed by Martin et al. [174] where they use a precision-recall framework [222]. Their method

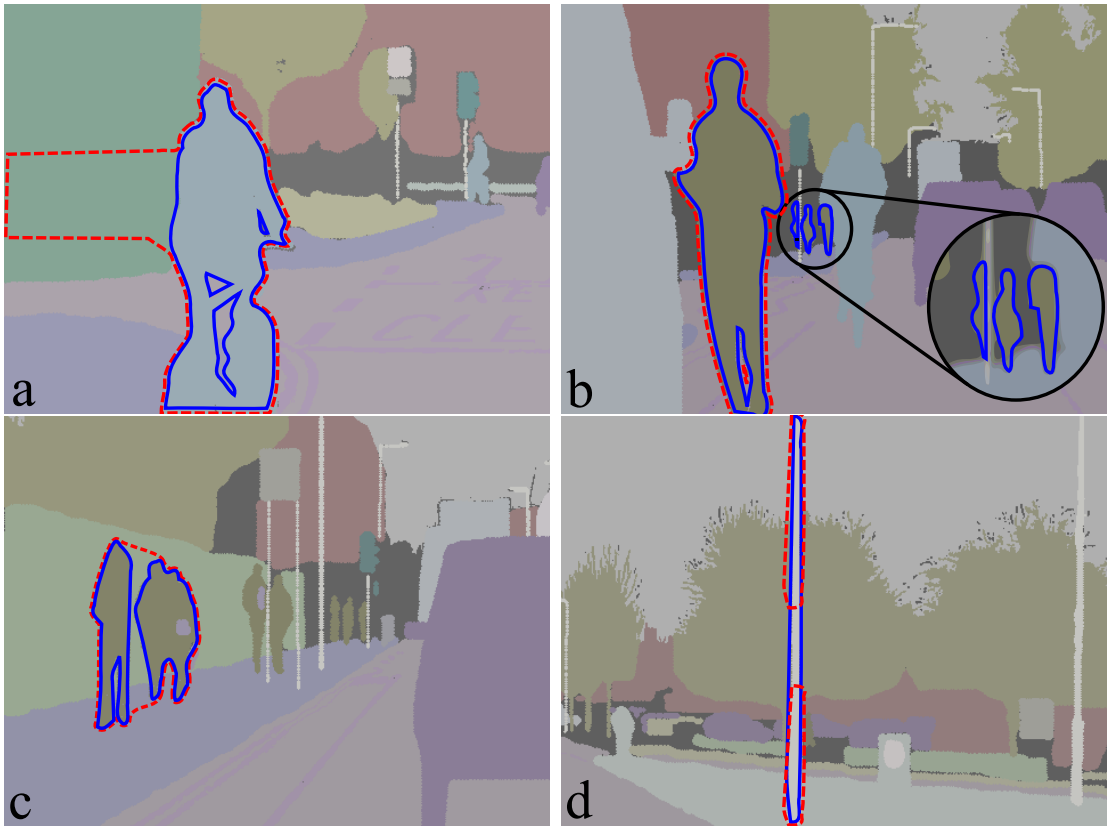


Figure 2.13: Segmentation vs. Detection. Blue outline for ground truth segments. Rad dashed outline for segmented regions. Four illustrative examples. a) Hit. Good detection but poor segmentation. b) Miss. Good segmentation but poor detection. c) Merge. Good segmentation but poor detection. d) Split. Good segmentation but poor detection.

balances how much true signal is required, R (recall), against how much noise can be tolerated, P (precision). The overall quality is summarized using the harmonic mean of precision and recall:

$$F = PR/(\alpha R + (1 - \alpha)P) \tag{2.11}$$

where α is usually set to 0.5 and they make use of the a bipartite graph matching algorithm for computing the hits and misses of boundary pixels. In the following section we demonstrate the differences in our datasets based on the reported performance of boundary detectors using this evaluation method. However, several failings of the boundary detection methodology proposed by [174] are highlighted in [188] and we will not use this measure in subsequent chapters.

2.5.3 Evaluation of Boundary Detectors

In Section 2.5.1 we made claims about the potential difficulty of the datasets based on the number of regions in the ground truth. Another way to demonstrate the difference is to empirically test competing algorithms on the datasets and highlight differences in performance.

To demonstrate the difficulty of the CamVid dataset we present boundary detection results from several papers in the literature [171, 221, 275]. Figure 2.14 presents precision-recall results for boundary detection on the BSD and CamVid datasets. We can see that the performance for `Canny` [43], `Pb` [174] and `BEL` [74] detectors is all considerably worse on the CamVid dataset.

One question then is: Should these differences in datasets be exploited by segmentation algorithms? We show in Chapter 4 that it is possible by learning a prior over the distribution boundaries of objects within a scene to improve segmentation performance.

2.6 Uses of Segmentation and Labelling

Section 2.4 reviewed several applications that are currently being examined in a research context. In this section we focus on technology that has made the transition from the research laboratory to commercial market place. Taking research from laboratory to market has important social and economic consequences but the transition is also vital for computer vision research because many techniques that have an elegant mathematical formulation or demonstrate impressive results in artificial conditions fail when they encounter real world examples.

Computer vision techniques have found application in a wide variety of industries including Medicine, Robotics, Film, Photography, Security, Transportation, Aerospace and Defence, and Science. One indicator of the broad industrial application can be seen by the number of multinational corporations that are actively involved in computer vision research, including: GE, Sony, Hitachi, Philips, Toshiba, Kitware, Microsoft, ObjectVideo, Honeywell, MERL, Cognex, Siemens, Vital Images, UtopiaCompression, Sarnoff Corporation, HP Lab, Adobe and Kodak. An illustration of selected products that use segmentation and labelling techniques can be seen in Figure **??**.

Segmentation techniques remain a key tool in aerial photographic mapping products and services, direct-digital macrophotography and topographic mapping for both

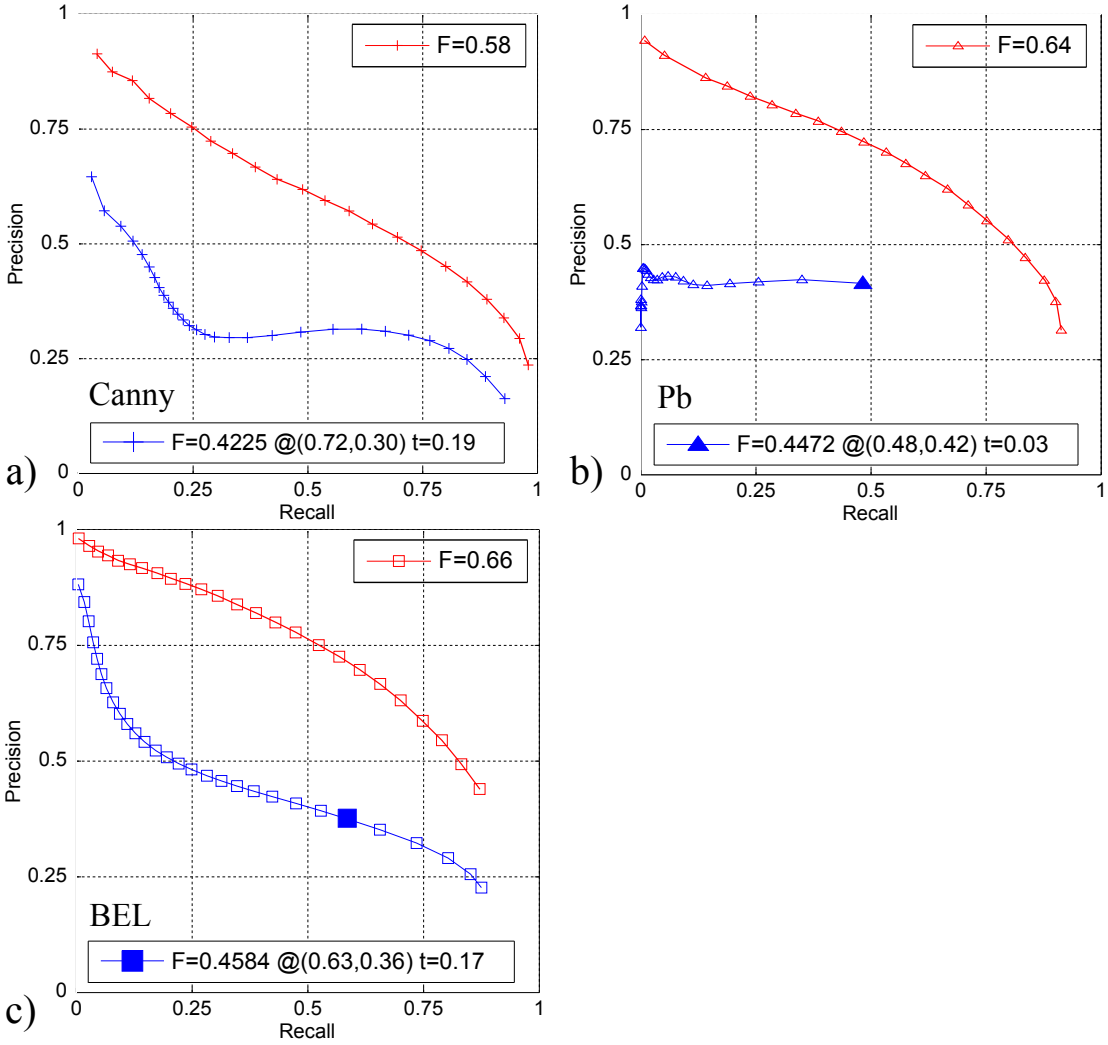


Figure 2.14: Precision Recall curves for competing boundary detectors on BSD (red) and CamVid (blue) datasets. This Figure compiles results from [171, 221, 275]. a) Canny detector [43] b) Pb detector [174] c) BEL detector [74].

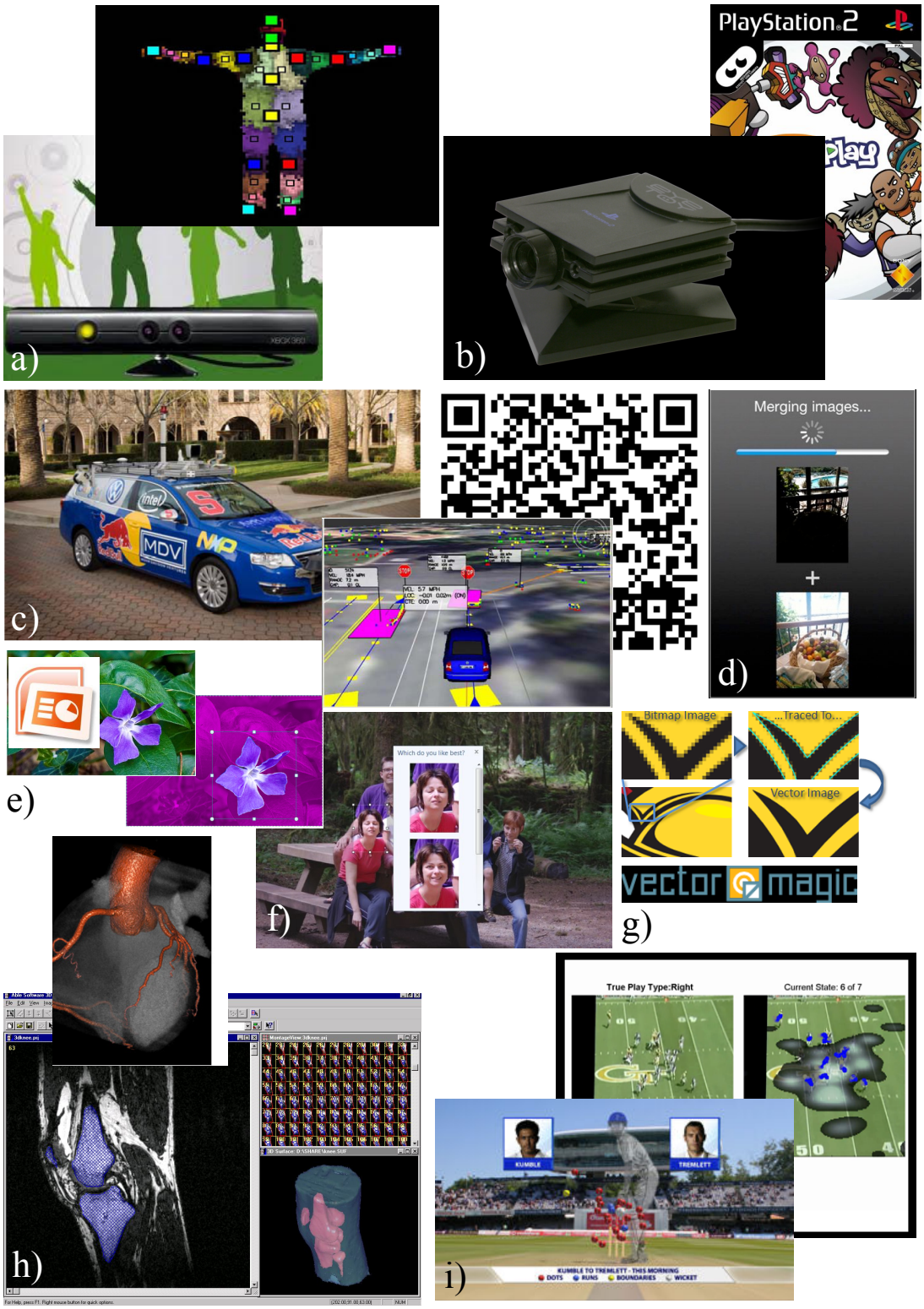


Figure 2.15: Exploitation. Recent applications of computer vision with focus on segmentation and labelling problems. a) Microsoft's Kinect camera interface and harlequin manikin [226]. b) Sony's EyeToy camera [223]. c) Standford's automated vehicle uses an MRF based algorithm. [253] d) Pro HDR mobile application [6]. e) Microsoft PowerPoint®[225] f) and PhotoFuse ®[12]. g) Automated raster bitmap format to vector svg using VectorMagic [7]. h) VitalImages Coronary CTA Analysis software [8]. i) Hawkeye Cricket officiating tools [3].

military and civilian applications [9]. Automated surveillance is also a growing sector with the increasing proliferation of camera systems. One example project is the use of ObjectVideo's real-time "first-response" system for the CCTVs used to monitor the Beijing Oympics 2008 [5]. Kitware's computer vision team is currently working with DARPA to develop the PerSEAS system for wide-area motion imagery analysis.

Recent medical imaging applications include Kitware's Slicer 4.0 project for medical image segmentation or VitalImages' Coronary CTA Analysis software for automated vessel segmentation and centerline discovery [8], see Figure **??**h. Image segmentation is also important for improving OCR recognition rates and compression efficiency within complex formats such as LEAD MRC, standard MRC T.44, and PDF and for converting between raster and vector image representations [7]. It is also has key applications in 1D and 2D barcode technology where inexpensive image-based readers are rapidly replacing laser scanners in a wide range of manufacturing and logistics applications [2].

There are several applications in the entertainment industry. Examples include video event and activity recognition using Kitware's VIRAT product [4]. Interestingly, another recent example is the application of HawkEye technology as an officiating tool in several sports. In Tennis the technology is an integral part of the ATP, WTA and ITF tennis tours, featuring at the Masters Cup in Shanghai, the US Open, the Australian Open, the Wimbledon Championships [3].

The commoditization of capture technology and the ease of distribution of images in high bandwidth networks has led to an increase in demand for image manipulation. Automated or assisted editing is now focus of many new applications and the new field of Computational Photography [194] which draws on many computer vision labelling techniques. Some applications include high dynamic range imaging, post-capture focusing, variable resolution, image relighting, and super-resolution. One recent example is an HDR application for mobile phones from eyeAppsLLC [6].

Microsoft™'s PowerPoint application is an example of consumer productivity tools that incorporate image editing tools based on computer vision research [225]. Labeling technology is also exploited in Microsoft™'s online Windows Live PhotoFuse product that allows users to create image montages by selecting different elements of several photographs [12].

Although not yet market-ready, there has been considerable progress on real-world applications of autonomous vehicles. One example, is the number of entries to the DARPA Urban Challenge that exploit computer vision techniques. An MRF formulation is used for segmenting lane markers in Stanford's autonomous vehicle [253].

Some of the most recent applications of computer vision and machine learning research have been in facilitating interaction with gaming consoles. Examples include the Sony EyeToy™[223] and Kinect from Microsoft's Project Natal™. The Natal project includes 3D tracking and motion estimation with semantic segmentation techniques [240] for real time human pose estimation [193]. This allows a user to interact with the console using gestures and body actions. The scope for this type of automatic body interaction offer many possibilities including hands-free control of devices during surgery or automatic fitting and bespoke tailoring in clothes industries [226].

Future application of the work included in the thesis might include foveated image and video coding. Another possibility would be interactive lattices - dividing images up into regions based on user input - similar to the use of seeds in user constrained segmentation. This may be useful if a user wants to impose a strict topology on the segmentation output [67].

2.7 Conclusions

In this chapter we have presented methods for both Segmentation and Labeling problems and shown many examples of superpixels being used in different computer vision tasks. Often, we have highlighted the limitations of a particular algorithm, for example that it is slow [238, 185] or susceptible to noisy estimates of data [89]. Additional comparisons of selected algorithms and further references are given throughout each chapter.

However, none of the methods presented control the topology of the resulting superpixel segmentations. Often restricted topology is a useful property and several applications presented in this review exploited regular sampling of pixels blocks [115, 112] or mapped superpixel segmentations without restricted topology back onto a regular grids for further processing [289, 290]. In Chapter 3 we argue that a number of desirable properties of pixels, that result from regular topology, should be maintained by superpixels and develop an algorithm that satisfies some of these additional constraints.

We also introduced several datasets that contain different modes of variation, including the number of objects in the ground truth data. We highlighted results in the literature that show very different performance of boundary detectors on two of these datasets and suggested that these difference might be exploited to improve the performance of algorithms. This is discussed at greater length in Chapter 4. We presented a framework for labelling problems, the Markov random field, that has proved very successful for supervised segmentation tasks. This framework is conspicuous by its absence in the list of methods presented in Table 2.1 that use superpixels. This motivates our work in Chapter 5.

Additionally, we have noted that progress in segmentation techniques has been rapid over recent years, both the application of supervised techniques [225, 12] and in the theoretical understanding of the relationship between competing techniques [241, 109]. However, the subject of segmentation is yet to achieve the broad utility of other "off the shelf" processing techniques that we encounter in standard image processing techniques like image coding and signal theory [70].

One goal for segmentation research must be to mature to a point where it can be used routinely in technology based on a sets of international standards in a similar manner to image coding and transmission problems eg. MPEG-1 H.264 [277]. The contributions of the thesis make progress towards that goal.

Chapter 3

Superpixel Lattices

In this chapter we discuss the motivations for superpixels in more detail. We argue that a number of desirable properties of pixels should be maintained by superpixels and that this is not possible with existing algorithms. We develop an algorithm that satisfies some of these additional properties and demonstrate performance on standard datasets.

3.1 Introduction

The principle of using image fragments to aid the process of inference has a long history in vision research. For instance, early papers using fragments for motion estimation include [82, 131, 279].

However extending the use of fragments to produce good unsupervised segmentations - where the algorithm does not use explicit information about object detection or recognition [139] - has proved challenging.

One reason for this is that the strategies that have proved successful in supervised segmentation are difficult to transfer in an unsupervised setting. For instance, in Chapter 2 we saw that one solution was to use human input - strong supervision - to set certain pixels, seeds, as *hard constraints* to guide the final segmentation. However, specifying these hard constraints in an unsupervised setting is difficult as we do not know which pixels belong to which objects in the scene. Even specifying these hard constraints in weakly supervised manner is difficult because it requires object detection and unfortunately detection rates on real world datasets remain low [87].

Alternatively, there are many approaches that use sources of top-down information [126, 33, 143, 282, 31, 157] to impose *soft constraints* to limit the segmentation to spe-

cific classes or shapes or sets of ordered parts. These methods produce useful results, but again this is harder to achieve in an unsupervised setting as specifying a model of a scene is difficult because of its varied composition. This means it is difficult to exploit the ordering constraints and fixed structure that are common to other top-down model driven segmentation methods and those that exist scale poorly as the number of classes increases [139].

Nonetheless, a review of the literature demonstrates the continued use of segmentation in many vision pipelines (see Table 2.1). One area of ongoing research in segmentation in an unsupervised setting has been motivated by work on *over-segmentation* where the image is divided into many small pieces that deliberately split the individual objects that comprise a scene into disjoint regions. These 'small fragments' have become colloquially known as *superpixels* and we examine the motivation for them in greater detail in the following section. We refer to algorithms that over-segment the image as *superpixel algorithms*.

3.1.1 Superpixels and the role of over-segmentation

Superpixels - literally the next thing 'above pixels' - were originally proposed in the paper of Ren and Malik [218]. They present them as a preprocessing stage in a vision pipeline and motivate this with two observations:

1. the number of pixels is high even at moderate resolutions; this makes optimization on the level of pixels intractable; and
2. pixels are not natural entities; they are merely a consequence of the discrete representation of images.

Ren and Malik suggest that a different representation of image data would be more appropriate for image reasoning tasks and define their concept of a *superpixel* as a "spatially-coherent, homogeneous, structure which preserves information over scales or sampling resolutions". We discuss these observations in greater detail.

We can re-state the first observation slightly to make the following assumptions clearer: 1) Inferring object labels at the pixel level is unnecessary in natural images because it is common for objects of interest to be large in the image and composed of many similar pixels. 2) Even accounting for progress on discrete optimization and the increasing speed and parallelism of hardware, inference can be slow in large images

when MRF or CRF graphs have one node for every pixel – and the size of captured images continues to increase. Their conclusion is that computational resources are wasted propagating redundant pixel information across the image and the solution they propose is to contract the number of nodes from a pixel graph to a superpixel graph. However, if the role of superpixels were solely one of efficiency, alternative solutions would include the use of standard multi-scale approximations [169], banded techniques [203] or novel coarse-to-fine message passing schemes [91] to reduce memory requirements and limit the complexity of solutions.

The second observation, that the pixel is a somewhat arbitrary unit for visual information, is more important and suggests a new representation. Recent work has demonstrated improved performance in recognition problems using restricted support [172] or using descriptors that accommodate local segmentation cues to produce *window-specific* features [199]. Window-specific features compute implicit 'soft-segmentations' of a sliding window region into foreground and background in an attempt to yield stronger object/background edges, and suppress textural and shading variations.

It is argued in [172] that restricted spatial support reduces the amount of noise present in a final feature vector, or descriptor, by virtue of the fact that it is based on pixels with similar properties. Furthermore, if pixels can be taken from the same object class it mitigates the effect that occluders have on diluting or masking a feature vector. An additional benefit might be that that as the amount of noise is reduced the descriptors themselves can be made simpler. The reason the use of superpixels is common in the literature is that they offer the promise of a data-driven region of support for feature vectors.

There are also several studies that suggest that the selection of features of *intermediate complexity* are optimal for classification tasks [263, 114]. While the notion of *intermediate* is difficult to define it is evident that it is possible to incorporate informative mid-level grouping cues such as curvilinear continuity [218] that are unavailable at the pixel level or regions with a uniform distribution.

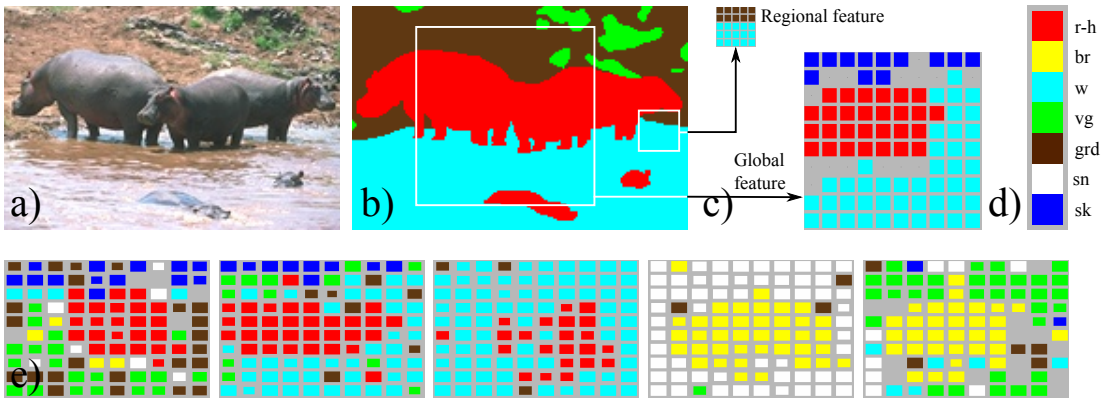


Figure 3.1: Illustration of learned regional *label features* from He et al. [115]. a) Original image. b) Example ground truth labels. c) Example distribution of labels at different scales in the image - regional and global. d) Labels from dataset. e) Five example global *label features* on 10×10 blocks. The area and colour of each square corresponds to the probability and dominant label at each site.

3.1.2 Motivating a new algorithm

The motivation for exploring a new superpixel algorithm came from a paper of He et al. [115]. In this work the authors propose a CRF approach to image labelling that includes contextual features at different scales in the image. Their system is composed of several different components each of which specializes on a different task. One such task is estimating the regional and global distribution of labels, learnt from training data. They call these *label features* and use them represent the context at a given scale by encoding a particular label pattern.

To represent label features they use sets of regularly sampled pixel sites that they refer to as *blocks*. Labels within these blocks of pixels are learnt at a fixed scale over regions of the image. Examples of these label distributions taken from their paper can be seen in Figure 3.1e. However, the authors note that:

> *"Ideally the system we described would be applied to a higher level of representation [than pixels]. However, this requires a consistent and reliable method for extracting such representations from images."*

In this statement He et al. [115] allude to the fact that the underlying graph that is used to represent an image tends to change significantly when it goes from one that represents pixels to one that represents regions or superpixels. This is illustrated in Figure 3.2. We begin with a regular sampling structure that represents the physical

arrays used to capture light in the imaging device (see Figure 3.2a). These arrays are used to produce a set of coloured pixels that are the most common representation of images (Figure 3.2b). Often this original pixel representation is altered to select or emphasize particular features that are of interest. For instance, Figure 3.2c shows one common operation - boundary detection [174]. These types of low level image operations like filtering or morphological alteration can be thought of as acting on the original pixel lattice.

However, when we begin to segment an image, as in Figure 3.2d, we almost always lose this regular graph structure. A graph representing this segmentation of the image can be seen in Figure 3.2h, where a single node, placed at the centroid of each region, is used to represent each segment and the edges between nodes represent the pixel adjacency of segments. The nodes in this type of graph can vary greatly in their connectivity and distribution within the image depending on the segmentation. In general, superpixel algorithms segment an image in a data dependent manner and the lower dimensional graph can have a very different connectivity and topology from the original graph used to represent pixels. It also means that there is no guarantee that there is any similarity between superpixel graphs that comprise a sets of images.

Why might this be important? One way to answer this question is to start by asking what are the useful or important properties of pixels:

1. Pixels can be represented in arrays without the need for pointers.

2. The uniformity of the graph it is easy to sub-sample pixels uniformly and use multi-scale methods.

3. The n^{th} pixel has a consistent, ordered, position in the image.

4. The n^{th} pixel has a consistent relationship with the $(n-1)^{th}$ pixel, allowing simple local neighborhood operations.

5. Pixel representations of images of the same dimension $(row \times col)$ are isomorphic which means there is a unique mapping from each pixel from one image to another.

These properties result from using a graph of regular topology to represent pixels.

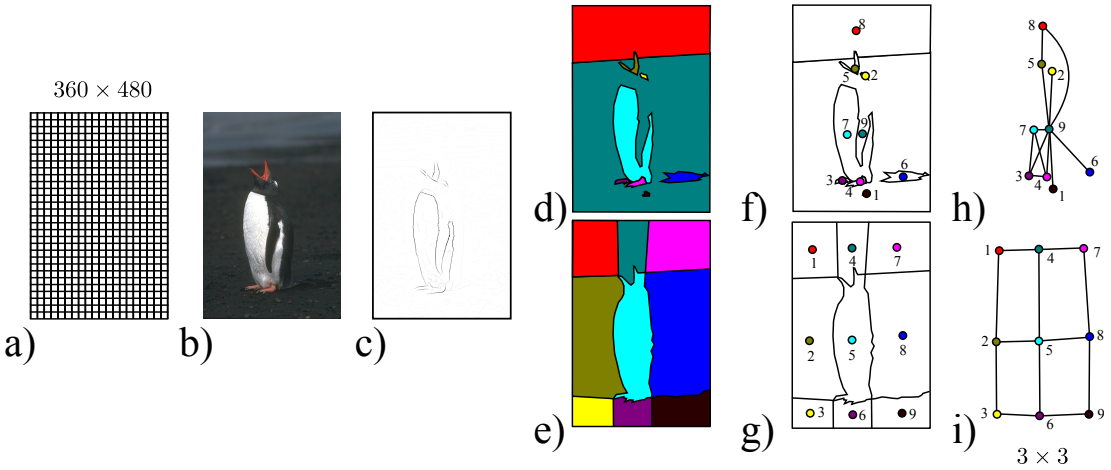


Figure 3.2: Representing images as graphs. a) Physical array used to sample incident light. b) Pixel representation of the image. c) Boundary map [174] d-f-h) Illustration of segmentation resulting in an irregular lattice. e-g-i) Illustration of segmentation with regular lattice. f-g) Nodes used to represent region properties placed at the centroid of each segment. h) Irregular graph with arbitrary topology and connectivity. i) Regular 3×3 graph where the nodes form a lattice.

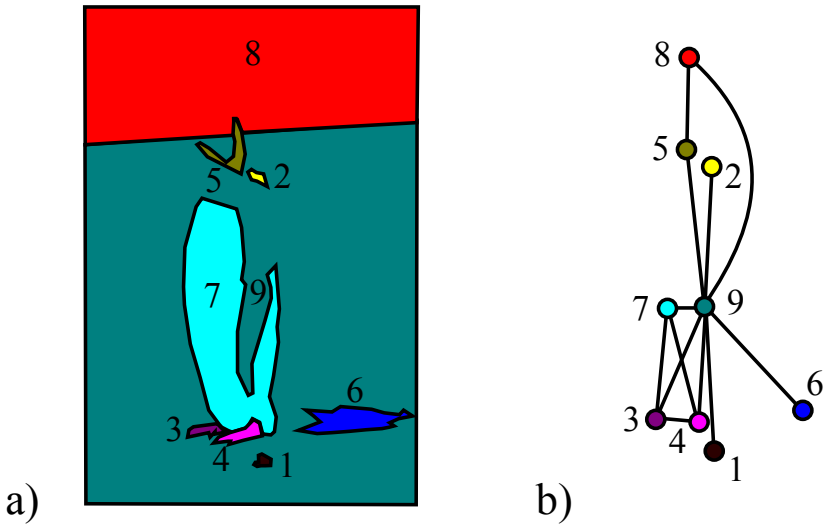


Figure 3.3: Problems of varying topology. Examples from Figure 3.2 shown in greater detail. a) Illustration of segmentation resulting in an irregular lattice. b) Irregular graph with arbitrary topology and connectivity. Note the different spatial arrangements between regions and the need to parameterize these differences on the edges of the graph.

These properties serve to highlight the advantages of working with regular graphs, some of which are simple engineering efficiency. These have practical consequences because regular graphs are easy to work with, can be stored in simple data structures and many algorithms designed for pixel grid graphs can be adapted for use in other regular graphs without the need to resort to more general graph algorithms [110].

Moreover, there are also several tasks that are hard to achieve in graphs with a non-regular topology. For example, it is not obvious how to learn the joint statistics of labels on graphs if the topology varies with each example. To illustrate this let us consider Figure 3.3. To learn potential functions on a particular graph that take account of spatial structure we would need to parameterize the set of edge and node relations or learn it from data. For instance, node 3 in Figure 3.3b is not really to the bottom left of node 9 but is on the edge of a region that encircles it on three sides (but not in the same way that node 1 is encircled). Another example is that we would like to take account of the fact the relationship between node 6 and node 2 is different from that of say node 5 and node 2 - although this is not obvious from looking at their nearest neighbours on the graph in Figure 3.2b. We are therefore left with questions as to whether we should we model different types of containment differently. In general it is usually difficult to characterize the spatial layout of segmentations due to their irregular shapes and arbitrary sizes [290].

In light of some of these difficulties researchers that want to exploit relative position tend to resort to either i) asymmetric potentials at the pixel level [283] ii) regular sampling of pixels blocks that are used to induce correlations indirectly [115, 112] or iii) mapping irregular segments back onto a regular grid graph [289, 290].

Our challenge then is to combine the concept of the superpixel suggested by Ren and Malik “a spatially-coherent, homogeneous, structure that preserves over scales” with the structural benefits we commonly associate with the representation of an image as a grid graph. This means producing a segmentation algorithm that captures some of the useful properties of pixels, listed in Table 3.1. In the following section we present an algorithm to do this - it constructs a regular lattice of superpixels - and we then demonstrate that it achieves reasonable performance when compared to competing segmentation methods.

Normal Pixel Properties		Common Superpixel Properties	
Fixed quantity	*Regular sampling grid with known dimensions*	Variable Quantity	*Variable sampling, data dependent.*
Ordered	*Fixed ordering between nodes.*	Unordered	*Ordering data dependent. Either decided at runtime or imposed by post-processing.*
Fixed Position	*Nodes always represent a region in the same physical location on an image.*	Variable Position	*Nodes can vary in spatial location.*
Compact	*Nodes represent constrained spatial regions.*	Extended	*Nodes represent regions that may vary hugely in size across the image.*
Multi-Scale	*Naturally multi-scale.*	Fixed Scale	*Algorithm dependent.*
Array Storage	*Fixed memory requirements.*	Pointer Referenced	*Dynamically assigned.*
Isomorphic	*Each node can be mapped uniquely from one graph to another*	Non Isomorphic	*Non exact matching between nodes and edges in separate graphs.*

Table 3.1: Pixel vs Superpixel properties. Comparison of pixel and superpixel properties for algorithms in the literature. Note that some superpixel algorithms are also naturally multi-scale [242].

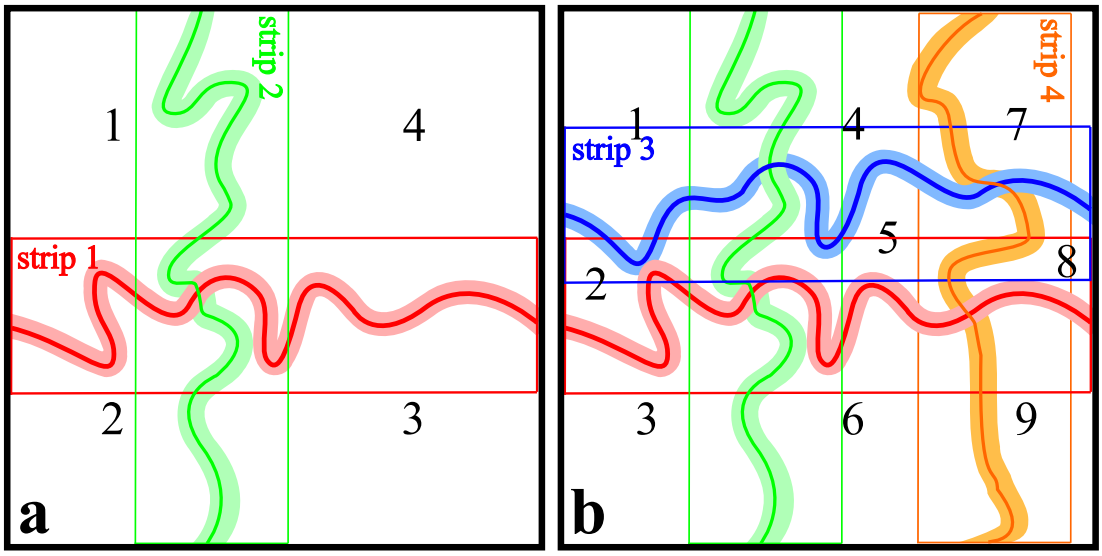


Figure 3.4: Incremental construction of superpixel lattice. a) The image is initially split left to right and top to bottom to form four regions. In each case we seek the optimal path within a predefined image strip. b) Adding one more vertical and horizontal path partitions the image into nine superpixels. Future path costs are modified in bands around previous paths (light colors) to prevent multiple-crossings and set a minimum distance of approach between paths.

3.2 Greedy Regular Lattices

We saw in Chapter 2 that most segmentation algorithms pose the question: "what properties would we like individual segments to have?", and develop different metrics for segment homogeneity. Here, we consider the question "what topological relations would we like to hold *between* adjacent segments?". In particular, a regular topology is our goal. We describe a greedy (divide and conquer) superpixel algorithm that maintains the regular topology of the grid graph of pixels.

We first present an overview of our approach and then fill in some of the detail for how we solve each subproblem in subsequent sub-sections.

3.2.1 Overview of our approach

The main idea behind our approach is that we divide the image up in to two sets of overlapping regions, or strips, that cross from one side of the image to the other in the horizontal and vertical directions. If we were to divide each strip in two down the middle and use this as our superpixel boundary then $N \times M$ strips would produce $(N+1)*(M+1)$ superpixels. However a straight path that divides each strip into two

would not preserve any object boundary information in the image.

Therefore, rather than divide each image strips evenly we seek a path from one end of the strip to another that conforms to object boundaries. To do this we use a shortest path algorithm in a graph where the edges represent the possibility that the pixel it represents belongs to an object boundary.

An example of what we want to achieve is illustrated in Figure 3.4. Initially, one path splits the image into two in the horizontal and vertical directions, to cumulatively produce four superpixels (see Figure 3.4a). At each subsequent step we add an additional vertical or horizontal path (Figure 3.4b). The series of strips act as a regularizing constraint that prevents the formation of paths that run diagonally across the whole image and hence restricts the placement of subsequent paths. It also forces a quasi-regular grid and reduces the computation at each step by limiting the number of paths considered.

Additionally, we alter the cost of edges in the graph around an existing path so that it sets a minimum distance of approach between parallel paths. This means that each superpixel is guaranteed to be a minimum width. An example of this is shown in (Figure 3.4b) for superpixel 2 where the blue path does not encroach on the light red band around the existing red path.

A regular lattice is guaranteed if we ensure two constraints:

1. PARALLEL: No two paths in adjacent strips of the same orientation cross each other.

2. ORTHOGONAL: Two paths in strips from different orientations cross exactly once.

We first describe two methods for solving the sub-problem of finding a path through each image strip, and then discuss how to ensure these constraints are maintained in 3.2.5.

3.2.2 Minimum Cost Paths

We model the image strip as a graph $G = \{V, E\}$ with a vertex set V corresponding to a node for each pixel in the image strip and an edge set E consisting of ordered pairs of vertices, $(u, v) \in E$, indicating the similarity between nodes. We use the term

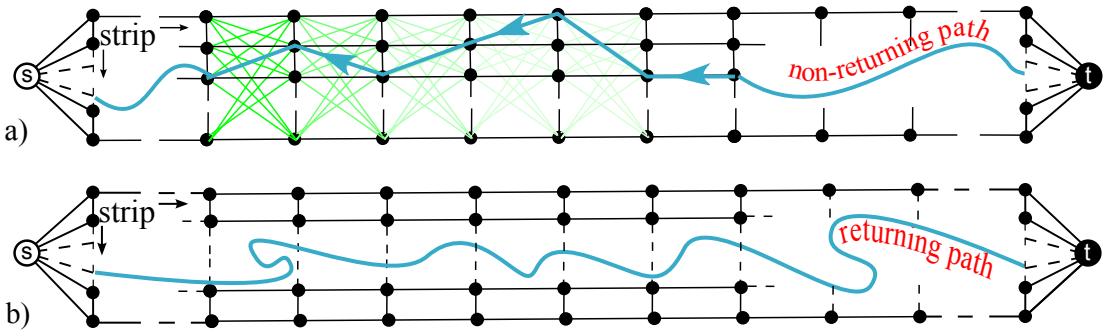


Figure 3.5: Estimation of optimal path through an image strip. a) Directed Acyclic Graph G_{dag}. Forward edges **green** are non-returning. b) Directed Grid Graph with positive edge weights G_{gg+}. This method is able to capture returning paths, ie. not every node on the paths is closer to the sink. Paths along the predecessor chain shown in **blue**.

boundary map to refer to a weight function $w : E \longrightarrow \mathcal{R}$ mapping edges to real-valued weights, $w(u, v)$. The mapping is represented by a 2D array containing a measure of the probability that a semantically meaningful boundary is present between two pixels. The details of setting edge weights and additional parameters are discussed further in Section 3.3.

The problem of estimating a *boundary map* is well studied in the literature. In the simplest case this can be the binary output of an edge detector [43] but in more complicated schemes leads to an estimate of the probability of *natural* [173, 74] or *occlusion* boundaries [124] in an image. These competing methods were discussed at greater length in Chapter 2. For convenience we invert and re-scale the boundary map to take a value of 0 where there is the most evidence for a boundary and 1 where there is no evidence. We term this the *boundary cost map* and discuss it further in section 3.3.

Our goal is to segment the images in places where this boundary cost map is lowest, which we do by finding minimum weighted paths through the graph. We therefore need to find the weight of different paths in the graph G where we define the weight of path $p = \langle \nu_0, \nu_1, \ldots, \nu_k \rangle$ as the sum of the weight of its constituent edges $w(p) = \sum_{i=1}^{k} w(\nu_{i-1}, \nu_i)$. We can state this minimum weighted path problem as a standard *single-pair shortest path problem* [54] with the addition of two special nodes to the graph, the *source* node s and *sink* node t. A single-pair shortest path problem means finding the minimum weighted path between these two nodes. The two addi-

tional nodes are attached to either end of the image strip and connected to all nodes representing pixels at the edge of each strip. For instance, we can see in Figure 3.5 that we are trying to find a minimum cost path from left to right in each strip.

We present two graph constructions and algorithms for finding shortest paths through the image strip. The first method, using a directed acyclic graph, produces paths that are *non-returning* such that every subsequent node on the path is closer to the sink node, see Figure 3.5a. The second method, using a directed grid graph, allows for *returning* paths of arbitrary shape, see Figure 3.5b.

3.2.3 Directed Acyclic Graph

The first graph we use to represent the pixels in a strip of the image is a directed acyclic graph (dag) $G_{dag} = \{V, E\}$. An example showing the graph for a 2×2 pixel horizonal image strip is shown in Figure 3.6a. A shortest path in this graph can be found with a well known *dynamic programming* routine [154].

All edges in this graph point towards a node that is closer to the sink. This means that an algorithm to find the shortest path can visit each node in *topological* order. The algorithm proceeds by visiting each node in turn and checking whether a path via this current node to an adjacent node is shorter than any previous path from other nodes already visited. If this is the case, then the cost of this path is updated so as to set a new lower estimate of the distance to this adjacent node, a process known as *relaxing* the edge on the path between the two nodes. After each node in the graph has been examined, and we therefore know the lower estimate of possible paths to the sink, it is possible to revisit nodes along the path with the shortest estimate to find the set of nodes that lie along the shortest path. The algorithm relies on the property of *optimal sub-structure* that is common to dynamic programming solutions. The optimal sub-structure of this problem is that a path between two nodes on a shortest path must also be a shortest path. We now describe some of the data-structures required for this algorithm more formally.

We require not only the cost of the shortest path but additionally a list of the vertices on the shortest path. We therefore maintain for each vertex $v \in V$ a predecessor $\pi[v]$ that is either another vertex or empty. To find the the set of vertices that make up the shortest path from t to s we simply traverse this predecessor list which runs in time

Algorithm 1 DAG-SHORTEST PATH (G_{dag}, ω, s)

```
1: INITIALIZE GRAPH (G,s) // Set estimate of upper bound on distance to ∞
2: for all u ∈ V, taken in topological sorted order do // Iterate through nodes in linear order
3:    for all v ∈ Adj[u] do // For every node adjacent to u
4:       if d[v] > d[u] + ω(u, v) then // Relax edge if new path is shorter
5:          d[v] ← d[u] + ω(u, v)
6:          π[v] ← u
7:       end if
8:    end for
9: end for
```

linear in the number of vertices in the path.

The second property we need to maintain for each vertex is an attribute $d[v]$ which is an upper bound on the weight (distance) of a shortest path from source s to v. The graph is initialized by setting an infinite upper bound on all nodes, excluding the source, and setting all predecessors with the empty set (see Figure 3.6a) - again time linear in the number of vertices.

The procedure for finding a shortest path involves the process of iteratively *relaxing* an edge (u, v). *Relaxing* an edge (lines 4-6 of Algorithm 1) means checking if it improves the estimate of the shortest path to v by passing through node u. If this new path improves the upper bound on the estimate of the distance to the node (ie. lowers it) then we update $\pi[v]$ and $d[v]$ accordingly. The algorithm for DAG-SHORTEST-PATHS(G, ω, s) is illustrated in Figure 3.6i)-vi). The upper bound on the current known distance $d[u]$ to the node is displayed in the middle of the node. Nodes that have been visited are coloured orange. The current update step is shown with a black node with red edges. The edges of predecessor nodes that are maintained in $\pi[u]$ are highlighted in blue.

DAG-SHORTEST-PATHS(G, ω, s) is shown in Algorithm 1, where an adjacency-list representation of the graph is used. For each $u \in V$, the adjacency list $Adj[u]$ contains all the vertices v such that there is an edge $(u, v) \in E$. A proof of the correctness can be found in Cormen et al. [54, pg. 593-594]. However our graph is constructed

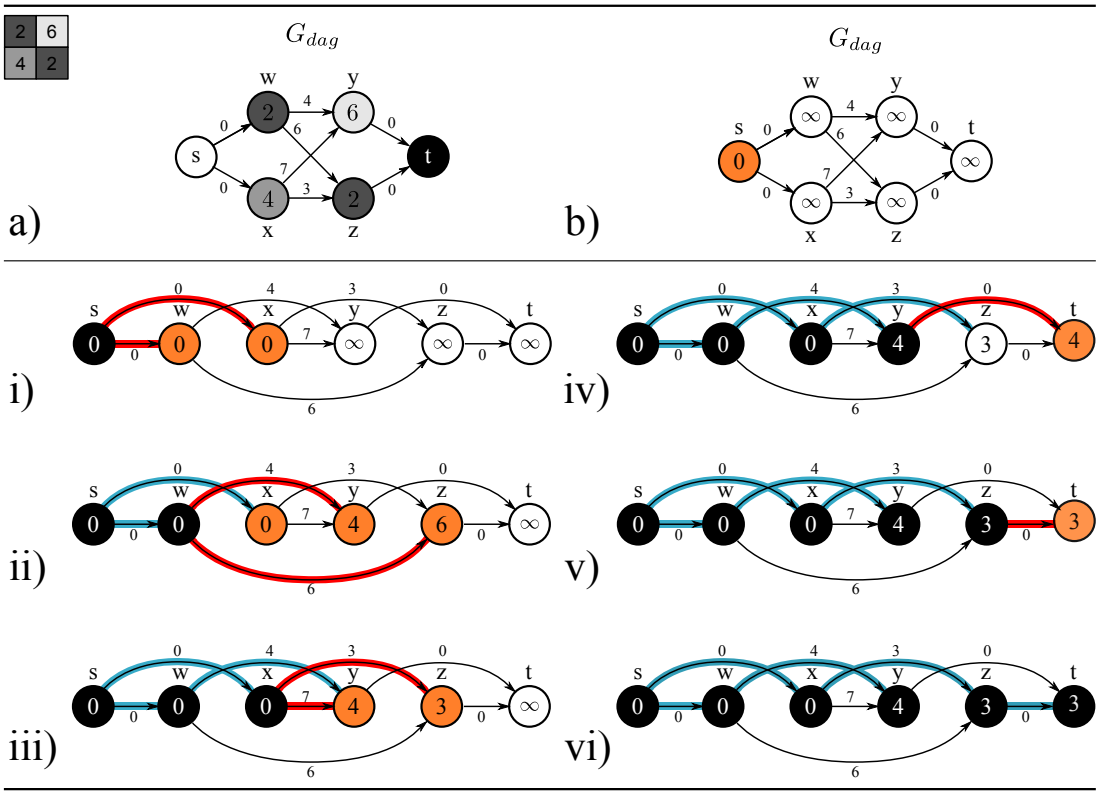


Figure 3.6: Graph construction and shortest path algorithm using a directed acyclic graph for a 2×2 horizontal strip of pixels represented by nodes $w - z$. a) Graph construction for G_{dag} where the weights are shown next to each edge. b) Initialization of the graph. i-vi) The algorithm proceeds by relaxing edges along an ordered set of nodes. Here we have redrawn the graph so that the nodes proceed in topological order from left to right using pixels in column-major order, ie. nodes are ordered in their distance away from the source. The upper bound on the current known distance $d[u]$ to the node is displayed in the middle of the node. Nodes that have been visited are coloured orange. The current update step is shown with a black node with red edges. Edges that connect to predecessor nodes that are maintained in $\pi[u]$ are highlighted in **blue**. In this simple example the minimum path is the same as Figure 3.7.

Algorithm 2 DIJKSTRA(G_{gg+}, ω, s)

1: INITIALIZE GRAPH (G,s) // Set estimate of upper bound on distance to ∞
2: $S \leftarrow \emptyset$
3: $Q \leftarrow V[G]$
4: **while** $Q \neq \emptyset$ **do**
5: $u \leftarrow$ EXTRACT-MIN(Q) // Get next node in queue
6: $S \leftarrow S \cup \{u\}$
7: **for all** $v \in Adj[u]$ **do** // For every node adjacent to u
8: **if** $d[v] > d[u] + \omega(u, v)$ **then** // Relax edge if new path is shorter
9: $d[v] \leftarrow d[u] + \omega(u, v)$
10: $\pi[v] \leftarrow u$
11: **end if**
12: **end for**
13: $Q = V - S$
14: **end while**

with an *a priori* ordering so we omit the usual topological sort of the vertices of G. The call INITIALIZE GRAPH in line 1 takes $\mathcal{O}(V)$. There is one iteration per vertex in lines 2-8 and each outgoing edge must be examined once giving a total of $|E|$ iterations lines 3-8. This gives a total run time of $\mathcal{O}(V + E)$.

3.2.4 Directed Grid Graph

The previous graph construction and shortest-path algorithm are fast, but the restriction on the shape of the path may limit the ability of the path to capture boundary information from the image. In the second graph construction therefore, we remove the restriction that each node on the path must be closer to the sink, and present another algorithm that can be used to find the shortest path in this new graph.

For our second graph construction we define a directed grid graph (gg) $G_{gg} = \{V, E\}$ over the image pixels in the strip as shown in Figure 3.7a. This graph construction contains loops which means we can no longer employ a *dynamic programming* method to solve the shortest path problem. However, if we impose the restriction that the edge weights are nonnegative (G_{gg+}), which is reasonable given that the edges ultimately represent probabilities, then we can solve the single-pair shortest path problem

with a well known greedy method known as Dijkstra's algorithm [72].

Dijkstra's algorithm maintains a set S of vertices whose final shortest-path weights from the source node have already been determined. The algorithm proceeds by examining a new vertex $u \in V - S$ with the current lowest upper bound $d[u]$, adds u to S and then relaxes all outgoing edges. Along with maintaining $d[v]$, $\pi[v]$ and S we must implement a min-priority queue Q of vertices, keyed by their d values to keep track of which nodes to visit next. This is illustrated in Figure 3.7i)-v), where the next node in the priority queue is coloured yellow. The algorithm for DIJKSTRA(G, ω, s) is given in Algorithm 2 and a proof of the correctness can be found in Cormen et al. [54, pg. 597-598].

The min-priority queue is initialized in line 3 of Algorithm 2 and maintaining this queue as the algorithm proceeds requires three priority-queue operations. The running time of Dijkstra's algorithm therefore depends on how the queue is implemented. If the data structure used to maintain the queue is a Fibonacci heap then we can achieve a running time of $\mathcal{O}(V \ln V + E)$.

Having presented two methods for finding minimum cost paths through strips that build up the lattice in a greedy manner we now discuss how to impose the constraints that guarantee the topology of the lattice.

3.2.5 Path constraints

Adjacent strips of the same orientation must overlap, to help preserve boundaries in the image, so this in itself is insufficient to prevent parallel paths from intersecting. Additionally, we need to ensure that perpendicular paths cross only once so that they do not create additional spurious superpixels in the image.

The result we would like to achieve for parallel paths (paths in adjacent strips of the same orientation) can be seen in Figure 3.8a. To do this we must limit the search space of each subsequent shortest-path to nodes that are not included in paths that have already been found. An example of this can be seen in Figure 3.8b. Here we have shown a path in strip 2 (red) that is limited to nodes in the purple and pink regions of the strip. The blue region is prohibited by the path found in the previous strip 1 (shown in white).

To achieve this first constraint we simply remove edges from nodes on existing

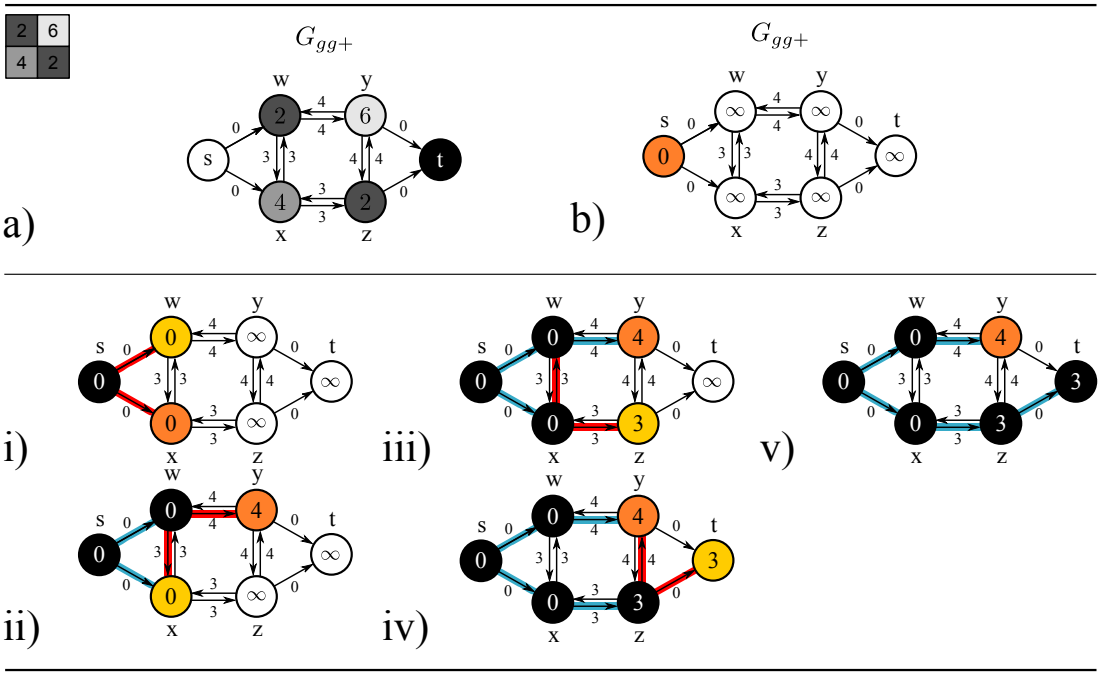


Figure 3.7: Graph construction and shortest path algorithm using a directed grid graph for a 2×2 horizontal strip of pixels represented by nodes $w - z$. a) Graph construction for G_{gg+} b) Initialization of the graph. i-iv) The algorithm proceeds by relaxing edges along the set of nodes maintained by min-priority queue, shown in yellow. The upper bound on the current known distance $d[u]$ to the node is displayed in the middle of the node. Nodes that have been visited are coloured orange. The current update step is shown with a black node with red edges. The edges of predecessor nodes that are maintained in $\pi[u]$ are highlighted in **blue**. In this simple example the minimum path is the same as Figure 3.6.

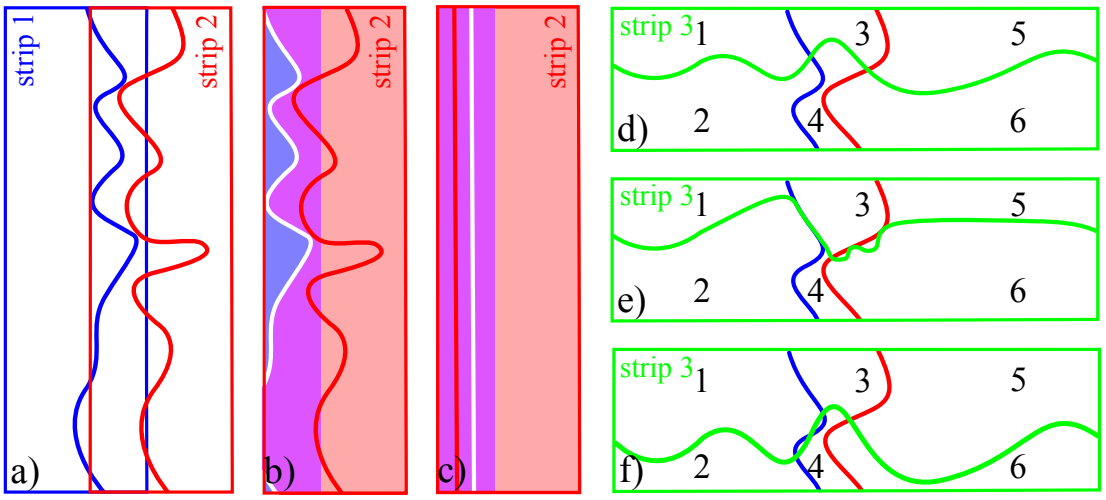


Figure 3.8: Example Path Crossings. a) Two parallel vertical paths in adjacent overlapping strips. The overlap means that the search space of the shortest paths overlap considerably. b) To prevent the path of the second strip (red) crossing the shortest path found in the blue strip we must remove edges from the white region. The region of nodes that appear in both searches, the strip overlap, are coloured purple. Nodes that are now un-reachable for a path in strip 2 are coloured blue. c) Example where two parallel paths occur in reverse order to the strips, ie. the path in strip 2 occurs to the left of the path (white) in strip 1. d) Orthogonal paths must cross only once. e) We must also prevent paths in orthogonal strips following the same boundaries. f) Lastly, we must ensure that no additional superpixels are created by paths crossing multiple times.

paths in the same orientation as the current strip. This prevents any possible shortest path passing from one side of a path to the other as there are no remaining edges.

Note that this parallel constraint does not force the order of the final paths. For instance, it would be possible that two parallel paths could be inserted in the overlap region of two adjoining strips in the opposite order to the strip themselves. An example of this is illustrated in Figure 3.8c where the new path (red) appears to the left of the old path (white) even though the new strip is to the right of the old one. However, this does not effect the final structure of the lattice which is governed by the number of paths, not their order.

The second constraint we would like to impose is that orthogonal paths cross only once. An example of the result we would like to achieve can be seen in Figure 3.8d. We must prevent a path in a new strip following existing orthogonal paths, see Figure 3.8e, so that a single superpixel cannot be split into parts that are separated. Additionally,

crossing once ensures there are not small bits of superpixels that need to be merged arbitrarily in order to preserve the total number of superpixels, see Figure 3.8f.

To achieve this goal we alter the weight of edges in the graph after the addition of each path - we refer to these altered edges as *constraint* edges. The idea is to pay a fixed large cost every time we traverse an existing path. This constant is the sum of all edge weights in the original graph, $\gamma = 1 + \sum_{(u,v)\in E} w_{uv}$ which means that it is greater than any possible existing path. This has the effect of limiting the set of feasible shortest paths on the new graph such that they will only include a minimal set of these constraint edges - i.e.one edge for each existing path it has to cross - and therefore only cross each orthogonal path once.

The set of edges that need to be altered depends on the local shape of the path at any given point and can be implemented in several different ways. The details of how to apply these constraint to each of our two graph constructions are given in Appendix A. However, the number of nodes that need to be visited to set these constraint edges is linear in the number of nodes in the path and it therefore does not increase the complexity of two algorithms we have presented.

3.3 Parameters

There are three important parameters that control the final lattice. First, the *resolution* determines the total number of superpixels, which is set by the number of paths. Second, the *overlap* of each image strip. Strips from the image should overlap so that a real-world boundary may be followed from one strip to the next using different paths. Third, the *tortuosity* of the path determines the degree to which the curve deviates from a straight line. We discuss each in turn.

Figure 3.9a-e gives an example of increasing the resolution of the lattice for a particular image. We can see from this that we would expect the lattice to capture more details of the original image as the resolution increases. In the limit that there are the same number of superpixels as pixels the lattice approximation would introduce no errors. There is therefore a trade-off between the total number of superpixels and the quality of the representation. This leads to the question “What number of superpixels provides a useful representation of the image?” We answer this question in several different ways, which serve to highlight a useful range for the number of superpixels.

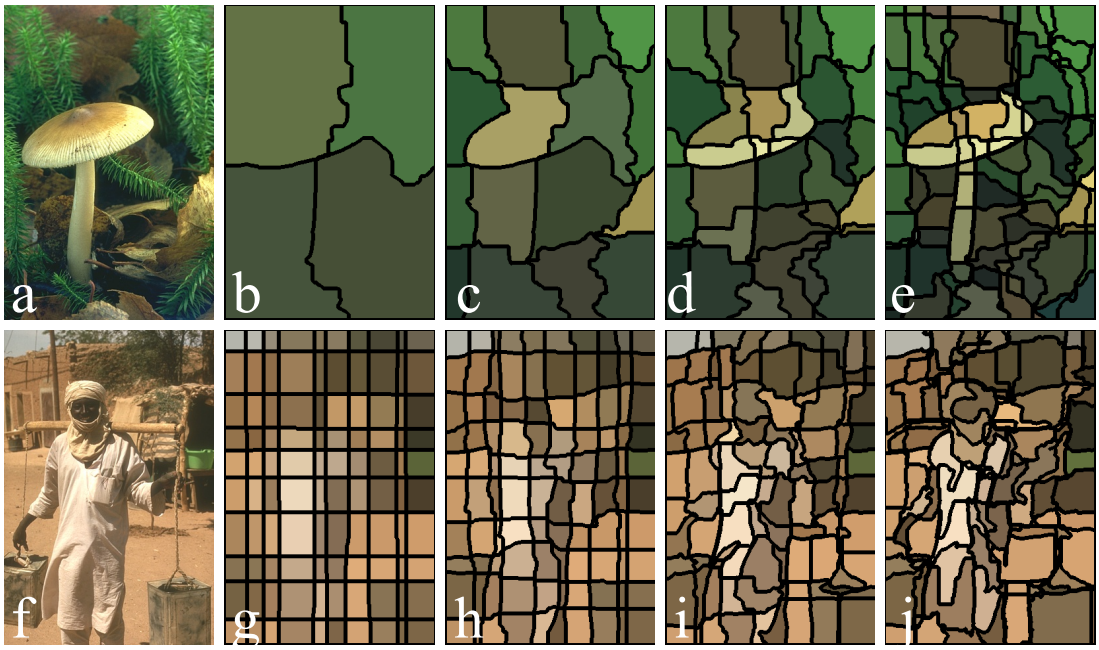


Figure 3.9: Two image sequences to demonstrate tuneable parameters of a greedy regular lattice. a) Original image. b)-e) Superpixel lattices with increasing superpixel resolution; 2×2, 4×4, 6×6 and 10×10, respectively. f) Original image. g)-j) 10×10 superpixels. Increasing tortuosity results in transition from straight grid to superpixels that conform to natural image boundaries.

The role of superpixels is to over-segment the image, usually segmenting objects of interest into several parts. Therefore, one way to gauge the useful number of superpixels is based on the number of regions of interest in a particular dataset. The datasets introduced in Chapter 2, BSD, VOC and CamVid, have on average approximately 20, 4 and 60 regions of interests respectively. In practice, we cannot provide a specific number of superpixels per region as this requires a solution to the problem we are trying to solve - identifying image regions. However, if there were an order of magnitude more superpixels than regions we might expect the useful number to be around 200, 40 and 600 respectively.

This estimate can be approached from the opposite direction by posing the problem as that of reducing the number of nodes used to represent the original image. For the BSD, VOC and CamVid datasets we have $\sim 154,000$, $\sim 178,000$ and $\sim 690,000$ pixels per image. If we were to reduce this by three orders of magnitude that would give a estimate of around ~ 150, ~ 200 and ~ 700 superpixels respectively, significantly increasing the efficiency of subsequent inference algorithms.

Ultimately, the range of useful superpixel segmentations should be determined empirically. However, it is possible that this would vary from application to application, and depend on the particular pipeline that is used during inference. For instance, there is some evidence that that the size of superpixel (image fragment) that provides optimal support for detection and recognition tasks is class specific [263].

In their seminal paper that introduced the notion of superpixels, Ren and Malik [218] explore the range $50-400$ and suggest that around 200 provides useful results on the BSD. They claim that when using only 200 superpixels the segmentation captures 90% of the human marked boundaries on the BSD and several hundred superpixel correlates well with previous heuristic estimates.

Using superpixels as regions of support for feature vectors was investigated by Hoiem et al. [121] in the first of a series of papers on geometric context from a single image. This paper exploited the use of around ~ 500 superpixels per image.

More recent work on using superpixels within a CRF framework by Gould et al. [105] investigate a range of superpixels from $100-800$ and demonstrate that performance of their system saturates in the region of 200 to 400 superpixels. Finally, work by Kohli et al. [139] on combining superpixels from multiple segmentations using a high-order CRF found that the range of $300-700$ superpixels provides good results. Table 3.2 shows a list of other pipeline applications, the superpixel algorithm with parameters and number of superpixels where they are made available.

There is therefore a range of the number of superpixels that are used experimentally in the existing literature and have been found to provide useful results. This number is commonly in the mid-hundreds. Pleasingly, this is a similar range to the number that might be expected from considering the role superpixels play within the vision pipeline where it is hoped we get a large compression ratio without too great a sacrifice in the segmentation quality. In our experimental evaluation we use a number of superpixels that span this range opting to explore lattice resolutions of 6×7, 10×10, 14×15, 18×19, 24×25 and 30×30 giving a total number of superpixels of 42, 100, 210, 342, 600 and 900. Other recent papers in the literature that investigate a similar range of superpixels are [11] and [159].

The second parameter, the overlap between strips, controls the width of each strip for a given number of paths. The greater the degree of overlap then the greater the

Pipeline Application	Method	#Superpixels	Parameters
Yang et al. [287]	[53]	< 450	$(h_s, h_r) = \{(7, 7)\}$
Kohli et al. [138]	[53]	$\sim 300-700$	$(h_s, h_r) = \{(7, 6.5), (7, 9.5), (7, 15)\}$
Ladicky et al. [148]	[53]	$\sim 150-920$	$(h_s, h_r) = \{(6, 5), (12, 10); (6, 7.5), (12, 15); (6, 9), (12, 18)\}$
Ren and Malik [218]	[238]	$50 - 400$	
Mori [185]	[238]	~ 1000	
He et al. [116]	[238]	~ 300	
Yang et al. [286]	[238]	$\sim 5 - 10$	
Cour and Shi [58]	[238][59]	60	
Rabinovich et al. [213]	[238]	$2 - 10$	
Rabinovich et al. [214]	[238]	$2 - 10$	
Sharon et al. [235]	[238]	~ 70	
Hoiem et al. [121]	[88]	~ 500	
Hoiem et al. [119]	[88]	~ 500	$\sigma = 0.5, k = 100, min = 100$
Russell et al. [227]	[88]	$\sim 2 - 15$	
Hoiem et al. [124]	[88]	$100 - 4400$	
Saxena et al. [230]	[88]	~ 2000	
Batra et al. [25]	[88]	$4 - 58$	$\sigma = 1, k = 150, min = 400$
Micusik and Kosecka [179]	[269]	~ 1000	
Malisiewicz and Efros [172]	[238]	$3 - 50$	
Gould et al. [105]	[238] [88]	$50 - 800$	
Pantofaru et al. [200]	[238]	$9 - 33$	
Todorovic and Ahuja [257]	[251]	~ 100	
Ahuja and Todorovic [15]	[251]	~ 50	
Warrell et al. [275]	[182]	25-625	
Galleguillos et al. [99]	[212]	20	
Fulkerson et al. [97]	[267]	$750 - 1200$	

Table 3.2: Pipeline applications of superpixels. Details of method, parameters and number of superpixels when made available in the publication. Numbers in **blue** are calculated independently.

search space afforded to each possible path. This increases the possibility of one existing path restricting the ability of subsequent paths to capture additional boundary information in the image. However, this ability to alter the search space is coupled very closely to the last parameter, tortuosity, so we examine this before showing experimental results for the joint parameter space in Figure 3.11.

The *tortuosity* parameter of the path helps determine the degree to which a path deviates from a straight line - or the extent to which path-length is moderated in our weight function $w(u, v)$. We first define a mapping which we call a *boundary map*

$$\beta(u, v) = T_1(p(u), p(v)) \tag{3.1}$$

which is a between pixel estimate of a boundary being present between node $p(u)$ and its neighbour $p(v)$ and convert it to a cost $\beta \in [0, 1]$. In practice we make T_1 a linear mapping, taking the average of the two probabilities: $\beta(u, v) = (p(u) + p(v))/2$.

We then use a second mapping that involves a parameter $\tau(u, v)$ based on the spatial relationship of the nodes u and v which we call the *boundary cost map*:

$$\mathcal{B}(u, v) = max(0, 1 - \beta(u, v) - \tau(u, v)) \tag{3.2}$$

where the $\mathcal{B} \in [2, 0]$ and $\tau \in [-1, 1]$. The effect of varying τ can be seen in Figure 3.9f-j. As τ decreases the path length cost of the minimum path begins to dominate and therefore the paths become straighter, ignoring the information in the boundary map. The parameter τ can take on negative values so the the cost of path length can be increased and results in straight superpixel boundaries.

An example of why this is important for the graph construction G_{gg+} is illustrated in Figure 3.10. Path-length means that paths that are in the direction of the sink are more favourable which means that that shortest-paths do not always follow boundaries in the image as we would wish. For instance, in Figure 3.10 the cost of the chord $a - b$ or $a - c$ depends on β. However, the choice of path along the chord $a - c$ or $a - d$ will always favour path $a - d$ based on path length. It does not take into account the cost of the edges on the path $c - d$ even though there is a strong boundary there. By subtracting τ from edges orthogonal to the direction of the strip we can help mitigate some of these path length effects.

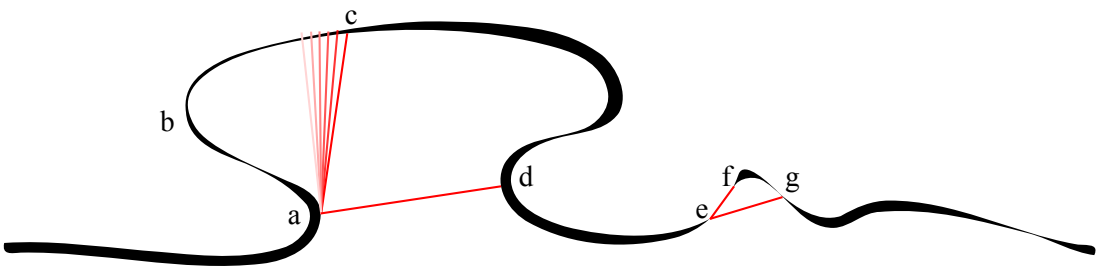


Figure 3.10: Tortuosity. Example horizontal path shown in black where the width of the line corresponds to the probability that there is a boundary present. The thicker the line the lower its cost in the *boundary cost map*. There are two costs associated with the boundary cost map. First, the mapping $\beta(u, v)$ turns the probability of a boundary into a weight in the graph. The second cost is that of edges in the direction perpendicular to direction of the strip. For instance, the path along the chord $a - d$ will always be preferable to the path along the chord $a - c$. To mitigate this path length effect we remove a fixed cost from edges orthogonal to the direction of the strip.

In addition to those parameters already discussed, we increase the edge costs of a band of nodes neighbouring each path to reduce the chance of paths becoming very close to one another (see Figure 3.4). This is undesirable because it can produce very small superpixels and, more importantly, close paths often follow the same real-world boundary, making the subsequent semantic interpretation of the intervening superpixels difficult [88].

Lastly, because we have a greedy algorithm based on finding paths through consecutive strips of an image the schedule that we use for the strips will alter the outcome. We sort the strips based on the values in the boundary map β so that strips with a greater chance of containing an object boundary are used first.

To demonstrate the effect of the parameters we conduct a simple experiment on the BSD training set of 200 images using the graph G_{gg+}. We use the `BEL` algorithm [74] (discussed in Chapter 2) to estimate the presence of an object boundary in the image and use this to set the boundary map β. To test different parameters we evaluate the performance of the algorithm by setting the j pixels assigned to the n^{th} superpixel to the mode class (most frequently occurring) of the ground truth data. This can be interpreted as using an ideal classifier.

Figure 3.11 shows the relationship between tortuosity and overlap for different lattice resolutions for five different performance metrics, introduced in Chapter 2. As the tortuosity of paths increase, they are slowly allowed to conform to the costs in the

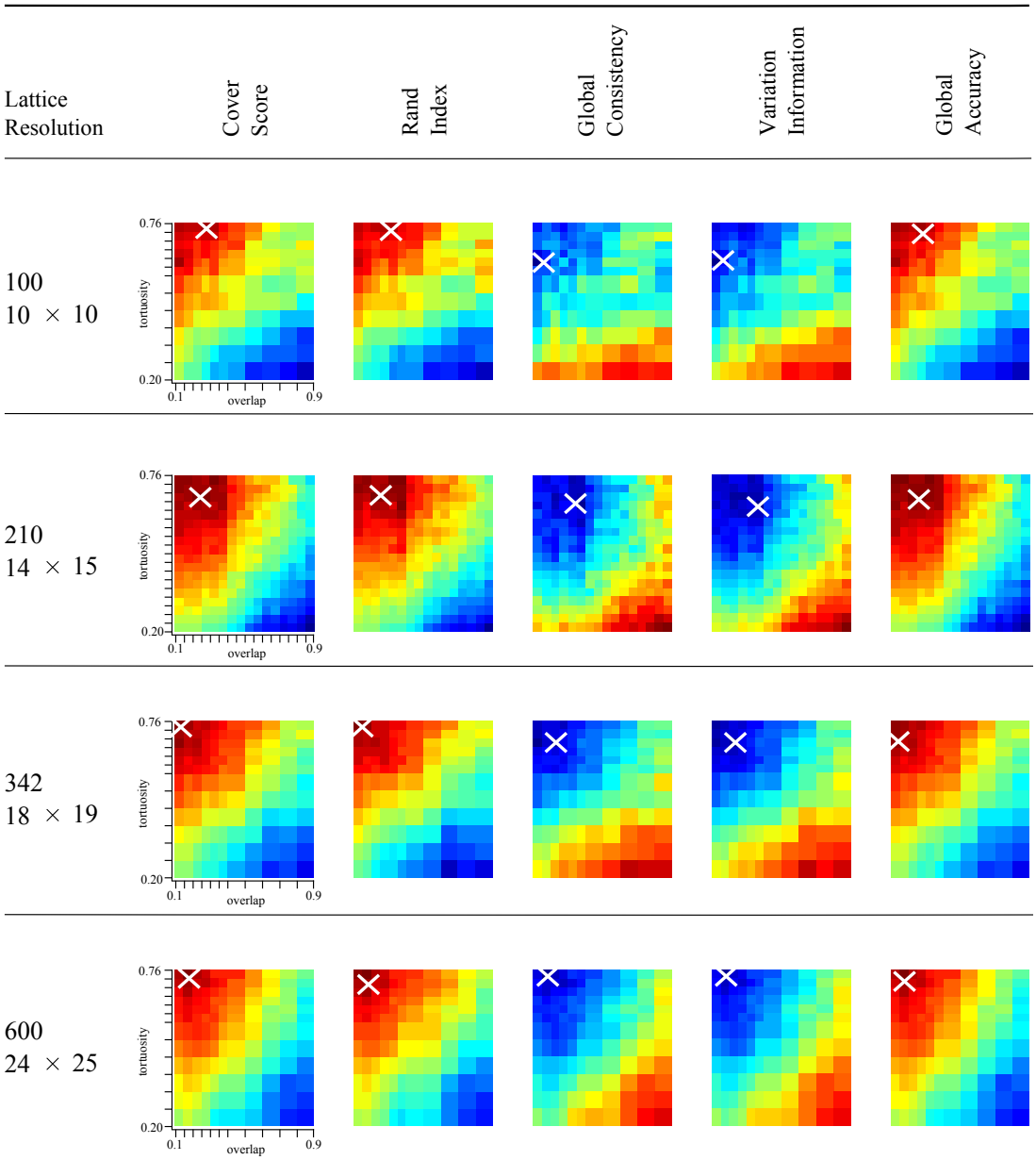


Figure 3.11: Exploring lattice parameters. Tortuosity vs Overlap. Plots of lattice resolution against five metrics of performance on the BSD training set with the BEL boundary map - with minima or maxima marked with a cross. Low tortuosity or large strip overlap reduce performance. The global extremum across both metrics and lattice resolutions are similar.

boundary map. Increasing this parameter will not always improve results because, at very high levels, the algorithm produces a meandering solution that attempts to assimilate all the image boundaries into a single path.

We can see in Figure 3.11 that the global extremum across different metrics and lattice resolutions are similar. Furthermore, each plot exhibits relatively smooth local structure which means that the algorithm is well behaved, i.e.small alterations in the parameters will not dramatically alter performance.

Averaging results across metrics and resolutions gives a tortuosity of 0.71 and an overlap of 0.17. This is helpful because if we use an overlap of $> 50\%$ of the strip width then there will always be a shortest path available regardless of the strip order. If, on the other hand the order is allowed to vary then we need the additional constraint that there is enough padding around the paths from neighbouring strips to ensure that there exists at least one shortest path in the center strip. As an overlap of 0.2 is used we need not consider this problem.

3.4 Qualitative Evaluation

In this section we give qualitative examples of the results of the algorithms before presenting quantitative results in Section 3.5. In Figure 3.13 we compare the regular lattice (top) to two other superpixel algorithms (middle and bottom) for the same image. The segmentations produced by our algorithm have a number of desirable properties that were highlighted in Table 3.1:

3.4.1 Consistent pixel positions

For a fixed resolution, each of our superpixels is always at roughly the same position in the image. This facilitates the definition of spatially varying priors over image classes as in [115]. For example, we can impose the information that superpixel 1 in the top-left of the image tends to be part of the sky. In other segmentation schemes, we would have to first establish the spatial position of superpixel 1 and then relate this to a spatial prior defined over the original image. This may be ambiguous if the region covers a large part of the image or contains peculiarities like holes. For example, is region 9 in Figure 3.3a at the bottom or in the middle? Often researchers that have exploited relative location priors between objects use regular grids even when working with superpixels [105] to

make this type of reasoning simpler.

3.4.2 Consistent spatial relations

In image parsing we want to learn the probabilistic relations between labels; for instance the frequency with which sky appears *above* the ground [116, 105]. While such relations can be defined *ad hoc* on any segmentation it results in a graph isomorphism problem: is the relationship between nodes in this graph the same as that encountered on other graphs during learning? A regular lattice means there is a bijection between segmentations (a one-to-one correspondence between segments), resulting in a consistent and unambiguous relationship between superpixels. This can be seen in Figure 3.13d. In contrast, in Figure 3.13e, which numbered region is to the left of region 244: 67,71 or 65? Which is under region 208: 73 or 67? Learning label distributions under this segmentation is ambiguous or involves imposing a new mapping. In practice, rather than trying to define the spatial relationship between superpixels by using centroids and angles of incidence several existing applications superimpose a regular grid, lattice, over the top of the segmentation algorithm [290, 105].

3.4.3 Scale Hierarchy

It is common to solve random field models using multi-scale techniques (e.g [203]). Regular lattices easily accommodate such methods as they have the same multi-scale relations as the original pixels: each superpixel decomposes into four smaller child superpixels. An example of this can be seen to happen between Figures 3.9b and 3.9c, again illustrated in Figure 3.12.

Figure 3.12: Scale Hierarchy.

3.4.4 Compactness

The segments in the superpixel lattice are *compact* [159], ie. of regular size and never greedily select huge image regions. This limits the possibility of erroneously grouping large semantically different regions such as sky and sea causing drastic ‘leaking’ between classes that can happen with other algorithms. An example large extended region is the sky in Figure 3.13c.

3.4.5 Graph Isomorphism

For a given resolution of lattice, superpixels in every segmentation have the same relationships with one another. This would allow the development of algorithms that learn of the relationships between the labelling of groups of superpixels (i.e.higher-order cliques). Without isomorphism, it is unclear how such cliques can be used to impose regular spatial information.

3.5 Quantitative Evaluation

In this section we demonstrate that our algorithm produces useful segmentations despite the added topological constraint of the lattice. Our evaluation is based on the three test datasets discussed in Chapter 2: Berkeley Segmentation Database (BSD) [173], Pascal Visual Object Classes (VOC) [87] and Cambridge-driving Labeled Video Database (CamVid) [39]. However, we first evaluate the performance of the two methods for constructing a superpixel lattice presented earlier in the chapter.

3.5.1 Comparison of competing graphs

We compare the performance of the graph G_{dag} and G_{gg+}. In a previous publication of this work [182] where the analysis of competing methods appeared we introduced an alternative metric for measuring performance, *explained variation*:

$$R^2 = \frac{\sum_i (\mu_i - \mu)^2}{\sum_i (x_i - \mu)^2} \tag{3.3}$$

where we sum over i pixels, x_i is the actual pixel value, μ is the global pixel mean and μ_i is the mean value of the pixels assigned to the superpixel that contains x_i. We saw in Chapter 2 that a metric like this is an an "empirical goodness" measure because it does not use ground truth data. Explained variation describes the proportion of image variation that is explained when the detail within the superpixels is removed.

The explained variation metric R^2 will take the maximum possible value of 1 as the number of superpixels increases and we recover the original pixels. It takes the minimum possible value of 0 when there is only one superpixel (the image mean). Our intention was to introduce a metric that was independent of human labeled ground truth and a similar measure of graylevel uniformity was independently introduced by Levine and Nazif [158]. The second metric we use, accuracy, is the number of true positive

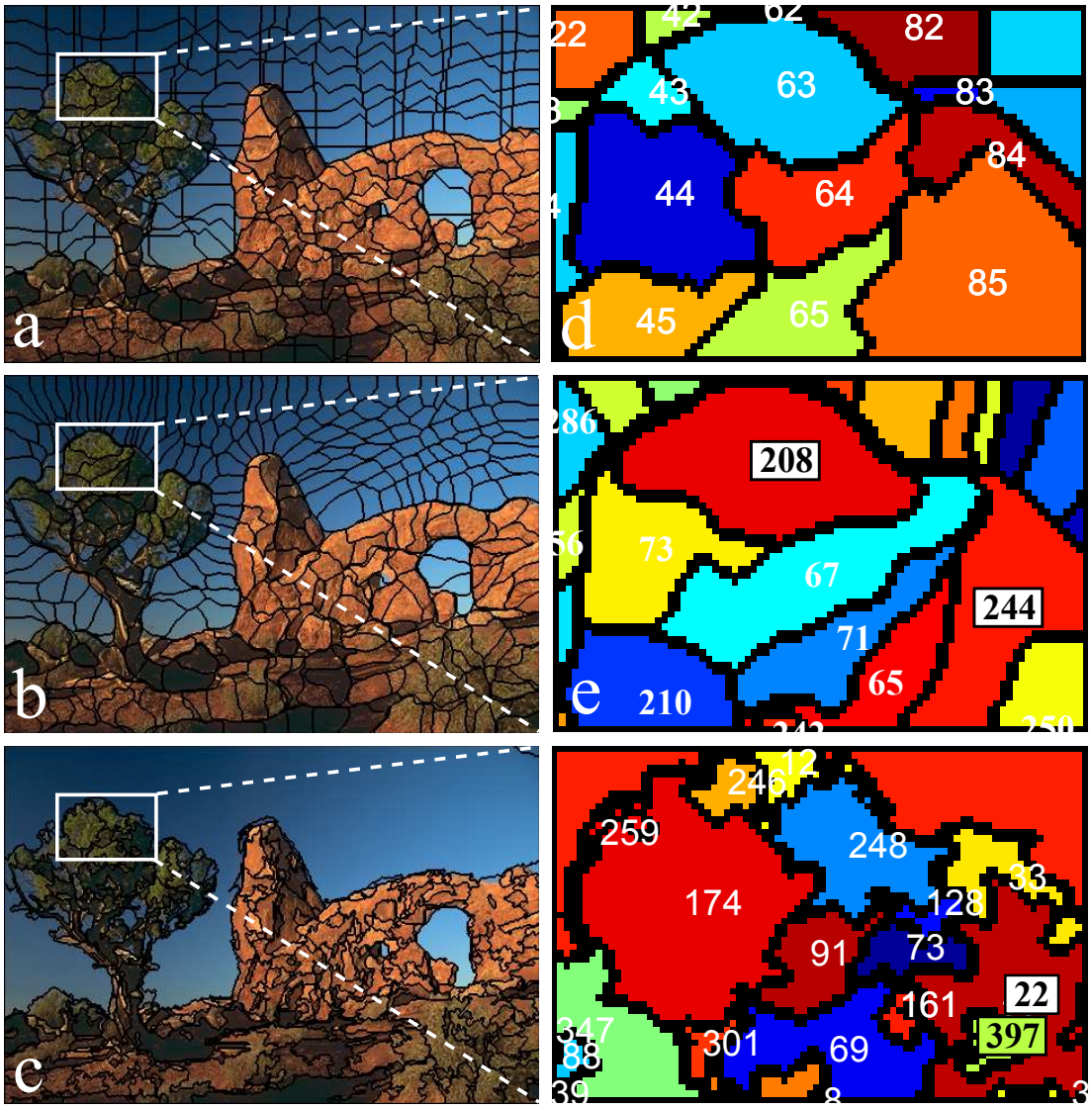


Figure 3.13: Comparing superpixel properties. a) Superpixel lattice method using graph construction G_{dag}. b) Normalized Cuts [238, 186] using the Pb boundary map [174] that is slow but provides convincing state-of-the-art performance against human labeled ground truth. c) Minimum spanning tree algorithm [88] that provides efficient segmentation benchmark. See Section 3.6 for detailed comparison of run time. d) Our algorithm maintains all the useful pixel properties that result from a regular lattice. e) This algorithm produces superpixels that have a variable number of neighbors in varying spatial configurations. This results in some ambiguous region properties. For instance, which numbered region is to the left of region 244: 67,71 or 65? Which is under region 208: 73 or 67? f) This algorithm additionally produces varying topologies. For example, superpixel 397 is completely inside superpixel 22.

Metric	R^2						Global Accuracy					
Lattice Resolution / Algorithm/ Parameters	42	100	210	342	600	900	42	100	210	342	600	900
	6×7	10×10	14×15	18×19	24×25	30×30	6×7	10×10	14×15	18×19	24×25	30×30
G_{dag} (Sobel)	0.40	0.63	0.69	0.72	0.74	0.76	0.72	0.88	0.91	0.93	0.94	0.95
G_{gg+} (Sobel)	**0.41**	**0.64**	**0.70**	**0.73**	**0.76**	**0.78**	**0.74**	0.88	0.91	0.93	0.94	0.95

Table 3.3: Performance of different graph constructions on BSD11 using explained variation and global accuracy.

and negative pixels over the total, see Appendix C.

Our evaluation is based on a selected subset of eleven images from the BSD, which we shall refer to as BSD11 (see Figure 3.14). These images were selected at uniform intervals from a list of the images ranked on performance of competing methods for boundary recognition tasks. This subset therefore represents a selection of both hard and easy examples and performance should be representative of the entire test dataset. The input to the transform β (Equation 3.1) is based on the Sobel operator for calculating image gradients on a 3×3 pixel region. Table 3.3 presents results for the two competing graph constructions.

We can see that the grid graph G_{gg+}, with fewer restrictions on the shape of possible paths, outperforms G_{dag} at almost every resolution. Better performance of the grid graph at lower resolutions is to be expected because there is greater separation between paths and therefore the added degree of freedom, returning paths, is likely have more benefit. There is a very slight decrease in performance at the highest resolutions using the metric of global accuracy. We interpret this result as the restriction on the paths in

Figure 3.14: BSD11 dataset. Images: id(rank): 42049(1), 189080(10), 182053(20), 295087(20), 271035(40), 143090(50), 208001(60), 167083(70), 54082(80), 58060(90), 8023(100)

Figure 3.15: Explained variation. Failure mode of the explained variation metric where performance improves as very small superpixels are encouraged in regions of constant texture with high pixel variance.

G_{dag} exhibiting slightly more stability in very weak boundary conditions, ie. wandering around less, but in practice the advantage is very small. An example using the directed acyclic graph is shown in Figure 3.13. Examples using the grid graph are shown in Figure 3.9.

However, the R^2 metric does not achieve exactly what we would want from a performance metric because it penalizes superpixels that contain consistent texture with large pixel variance. An example illustrating this problem can be in Figure 3.15 where we use the the algorithm of Felzenszwalb and Huttenlocher [88] because it exaggerates the effect visually. If we compare Figure 3.15b and c we can see that as the number of superpixels increases regions of texture with high pixel variance, eg. the front of the bird, are modeled by very small superpixels - sometimes even single pixels. This is not desirable because we may be interested in modeling classes with high variance texture with the same number of regions as more homogeneous texture, ie. having an equal number of superpixels for each region of interest. However, the *compactness* restriction of the lattice means that it is still suitable for evaluating our competing methods as neither can produce arbitrarily small regions in one particular part of the image. Therefore, while we use R^2 for analysis of the two methods we present, we will not use this metric further for evaluating competing superpixel algorithms.

The promising performance of G_{gg+} means we shall use this graph construction for detailed analysis against competing methods and choice of boundary maps. From now on we shall refer to this method as the *greedy regular lattice* or GRL.

3.5.2 Evaluation methodology

We follow a similar strategy to Everingham et al. [84] and use six measures for comparing the performance of superpixel algorithms: F_{sd} [183], Cover Score [19], Rand Index [216], Global Consistency Error[173], Variation of Information [178] and Accuracy. The methodology was introduced in the previous chapter (Section 2.5.2). In addition we follow [84] by using the number of superpixels and algorithm runtime as cost functions.

To measure over-segmentation performance we set the j pixels assigned to the n^{th} superpixel to the mode class (most frequently occurring) of the ground truth data. This can be interpreted as using an ideal classifier to label the set of pixels within a superpixel based on the dominant class and is similar to the notion of *potential accuracy* adopted in [84]. It should be remembered that because the segmentations being compared are based on a mapping of the ground truth data the measures appear unusually high (see Appendix B).

We are required to compare competing methods that do not all return a fixed number of superpixels. We therefore interpolate values at particular resolutions of superpixels using robust linear regression with iteratively reweighed least squares [125], using `robustfit.m` and the default weighting function. We use a piecewise linear approximation using data points in a band that includes the superpixel resolution below and above that being estimated. The resolutions we use were chosen to be approximately evenly spaced in each dimension of the lattice (eg. 6, 10, 14 ...). This produces a non-uniform sampling of the total number of superpixels, with a greater number of samples at lower resolutions. We note that it has the advantage that there are more samples where the gradient changes are greatest (eg. see Figure 5.15).

The interpolation procedure is illustrated in Figure 3.16. The regression line is shown in black and the standard error based on the regression is shown with red bars. Examples of the different distributions of points in the fitness/cost space for the `FH`, `MS`, `NC(Pb)` and `GRL(BEL)` algorithms and three different measures on the CamVid dataset can be seen in Figure 3.19.

It is possible for different algorithms to be dominant at different points in the fitness/cost space. An illustration of this is shown in Figure 3.17 where we show a fitness measure, Cover Score, against a cost measure, the number of superpixels, for

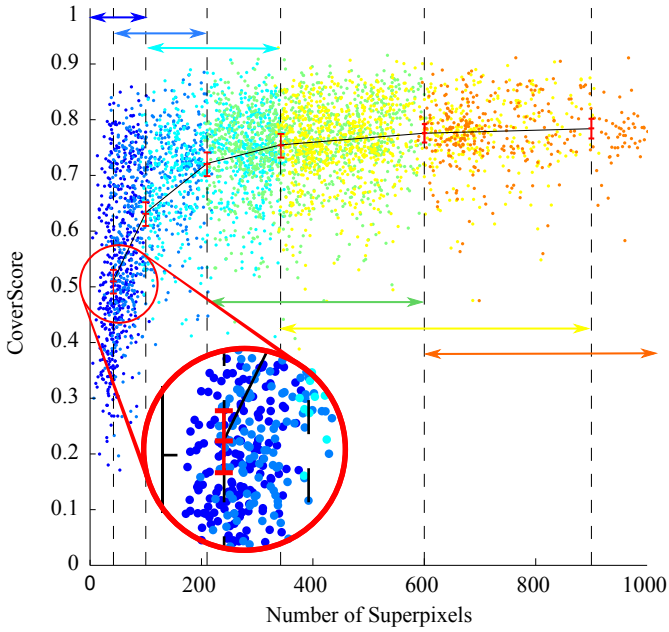
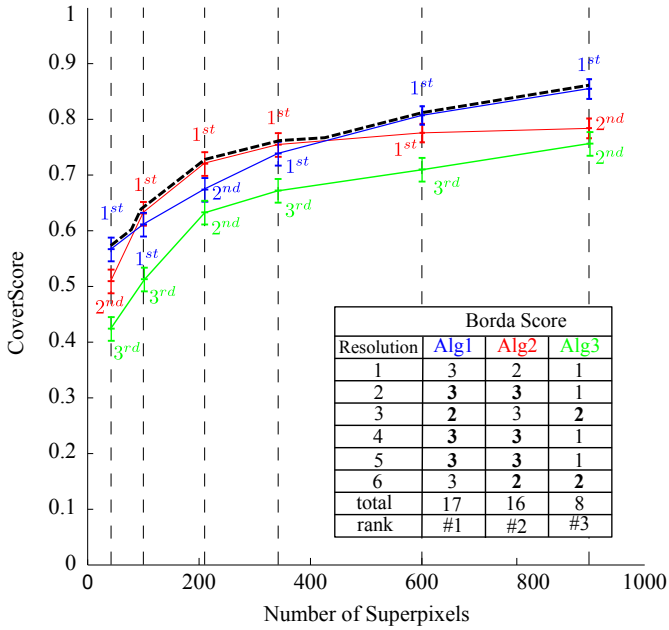


Figure 3.16: Robust interpolation. Example data for Number of superpixel vs Cover Scores. Each dot represents a segmentation for one image and given set of parameters. The piecewise linear approximation of algorithm performance (black) is interpolated from images containing superpixels in set coloured band either side of the desired resolution. a-inset) Inset shows position of mean and standard error for regression estimate at 42 superpixels.



Borda Score			
Resolution	Alg1	Alg2	Alg3
1	3	2	1
2	**3**	**3**	1
3	**2**	3	**2**
4	**3**	**3**	1
5	**3**	**3**	1
6	3	**2**	**2**
total	17	16	8
rank	#1	#2	#3

Figure 3.17: Algorithm ranking using Borda score of three illustrative algorithms. Partial ranking, ie. ranking with ties, is calculated for each resolution using the mean and standard error. The maximum rank contributes to the Borda score and ties have been highlighted in bold. The monotonic hull of this projection is shown with a dashed black line.

three illustrative algorithms. It is clear that Algorithm 3 performs worst as it never dominates any of the other Algorithms. However determining which of Algorithm 1 or 2 is more useful depends on a particular task or application.

Moreover, we would like to take account of the uncertainty of estimating the trend line using regression. To achieve this we score the algorithms based on their maximum rank. By maximum rank we mean the highest possible position of an algorithm at a point in the fitness/cost space given an estimate of its mean and standard error. An example of this is shown in Figure 3.17 where the maximum ranked position of the algorithm is shown next to the trend line. If the standard errors of two algorithms overlap we consider their performance to be indistinguishable and give them both the highest possible rank. For example, at the second resolution Algorithms 1 and 2 are both ranked first and Algorithm 3 third. Similarly, at the highest resolution Algorithm 1 is ranked first and Algorithm 2 and 3 are ranked second.

This procedure gives us a partial ranking (ie. a set of lists of the position of an algorithm that can contains ties) for each resolution, measure of performance and dataset. However, it is useful to have a single measure that can be used to summarize performance of competing algorithms. We therefore adopt a simple heuristic score known as the Borda score [63] which gives a score to an algorithm inverse to its position in a list. For example, consider the first resolution in Figure 3.17. Algorithm 1 is given a score of 3, Algorithm 2 a score of 2 and Algorithm 3 and score of 1. We arrive at a total score by summing up over resolutions (given at the bottom). Algorithm 1 has the highest total score by virtue of having more first ranked positions.

Partial rank aggregation allows us to summarize all the results that are presented in Appendix B in one simple number and is also used in [56]. However, it should be noted that our method has the disadvantage that it produces relative scores ie. if we run the analysis with a different number of algorithms then the Borda score changes for each algorithm. (For instance you cannot compare the scores for `GRL` from Section 3.5.3 and Section 3.5.4). Additionally, it does not give a measure of how poor performance is for lower ranked algorithms. Despite this, it gives a useful way for us to compare performance across multiple measures and datasets.

Algorithm	Reference	Datasets			
		BSD [173]	**VOC** [87]	**CamVid** [39]	Total
GRL (BEL)	[182] ([74])	**108**	**108**	**108**	**324**
GRL (Can)	[182] ([43])	88	97	101	286
GRL (Pb)	[182] ([174])	105	103	103	311

Table 3.4: Boundary map ranking using GRL algorithm with different boundary detection algorithms. Borda scores shown for three datasets summed across both metrics and resolutions. A total score across datasets is given in the last column. BEL algorithm is ranked first on all datasets. Maximum Borda score on individual dataset = 108.

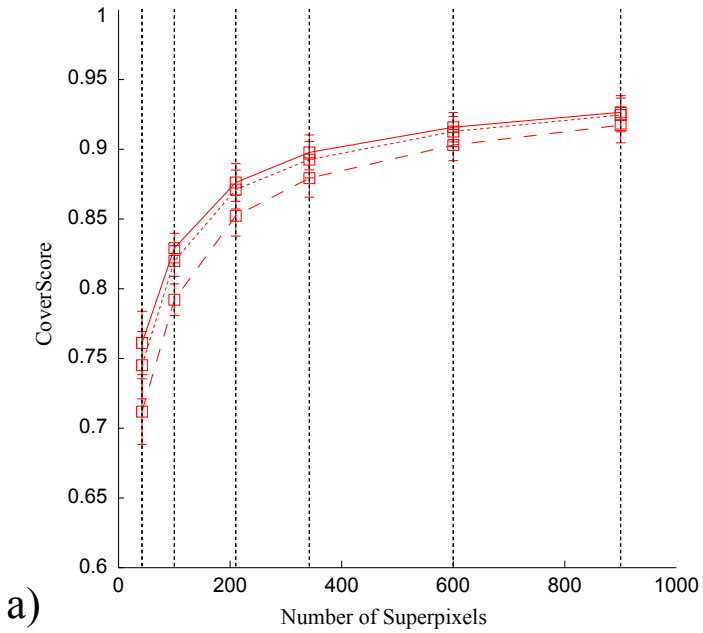
3.5.3 Comparing Boundary Maps

We investigate three choices of boundary map: The `Pb` boundary map [174] generated using gradient/texton cue integration that provides good performance against human labelling. The fast `BEL` boundary map [74] generated using a boosted classifier on a Canny edge mask to learn natural boundaries. This algorithm is efficient and produces a good precision-recall f-measure score [173] on the BSD and CamVid [274]. We contrast with a simple edge map generated using the Canny edge detector implemented by Matlab™.

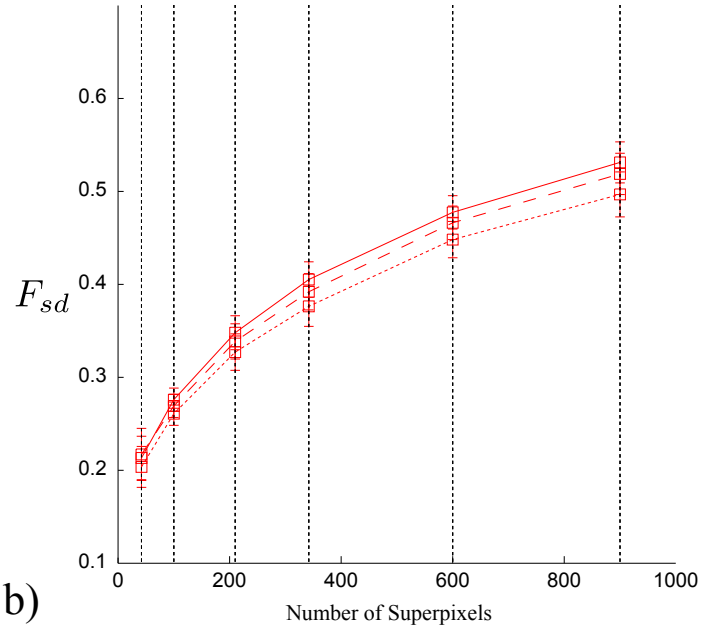
Full results using the Borda score can be seen in Table 3.4. We can see that the `BEL` algorithm is ranked first across all datasets but that there is not much difference between `BEL` and `Pb` overall. However the `BEL` algorithm is approximately an order of magnitude faster than `Pb` so it is more practical as a pre-processing step.

Plots of two measures on different datasets can be seen in Figure 5.15. For all boundary maps the performance improves as the number of superpixels increases, as expected. Note that the ranking for the F_{sd} measure (Figure 5.15b) is not the same as the overall ranking based on all measures. As the F_{sd} measure incorporates detection and segmentation separately it is particularly sensitive to missing small objects.

We can see the value of a ranking score across different measures as `Pb` remains ranked second overall, suggesting a useful performance gain over the Canny detector. This corroborates the results on boundary detection evaluation on the CamVid dataset presented at the end of Chapter 2 where the Canny detector had a boundary F-measure



a)



b)

KEY: Boundary maps: — BEL [74], - - - Pb [174], – – Canny [43]

Figure 3.18: Comparing performance of GRL using different boundary maps. a) Number of superpixel vs Cover Score on BSD [174]. BEL algorithm dominates other boundary maps. b) Number of superpixel vs F_{sd} on CamVid [39]. BEL algorithm dominates other boundary maps.

Algorithm	Reference	Datasets			
		BSD [173]	**VOC** [87]	**CamVid** [39]	Total
FH	[88]	114	131	125	370
GRL (BEL)	[182] ([74])	149	159	159	467
MS	[53]	154	159	122	435
NC (Pb)	[186] ([174])	**175**	**170**	**173**	**518**
uniform		95	100	129	324

Table 3.5: Algorithm ranking using Borda score on three datasets and combined totals. The NC algorithm clearly produces the best performance across datasets. However, our GRL algorithm performs well and produces the second highest total score. Maximum Borda score on individual dataset = 180.

score of 0.42 compared to the Pb score of 0.45 on the CamVid dataset. Based on these results we will use the BEL boundary map when comparing the GRL algorithm to competing methods in the rest of the chapter.

3.5.4 Comparison to other algorithms

We now compare our algorithm (GRL) to three other methods that are common in the literature: Normalized Cuts (NC) using an implementation made available by Mori [185], the agglomerative method of Felzenszwalb and Huttenlocher [88] based on the minimum spanning tree (FH) and the Mean Shift (MS) algorithm of Comaniciu and Meer [53].

These algorithms represent either ends of a spectrum of segmentation algorithms: the NC algorithm provides state-of-the-art performance against human-labeled ground truth data when using the Pb boundary map [19, 184]. The method tends to be too slow to be a viable preprocessing component of a vision algorithm pipeline (even multi-scale implementations are in the order of minutes for reasonably sized images [59]) but provides a useful upper bound on segmentation performance.

The MS and FH algorithms set a benchmark for efficiency and are commonly used in practical applications, see Table 2.1. We use the EDISON implementation of the MS algorithm made available by Christoudias et al. [52] including the 'synergistic' inclusion of gradient information and region post-processing steps. We perform no post-processing on our algorithm.

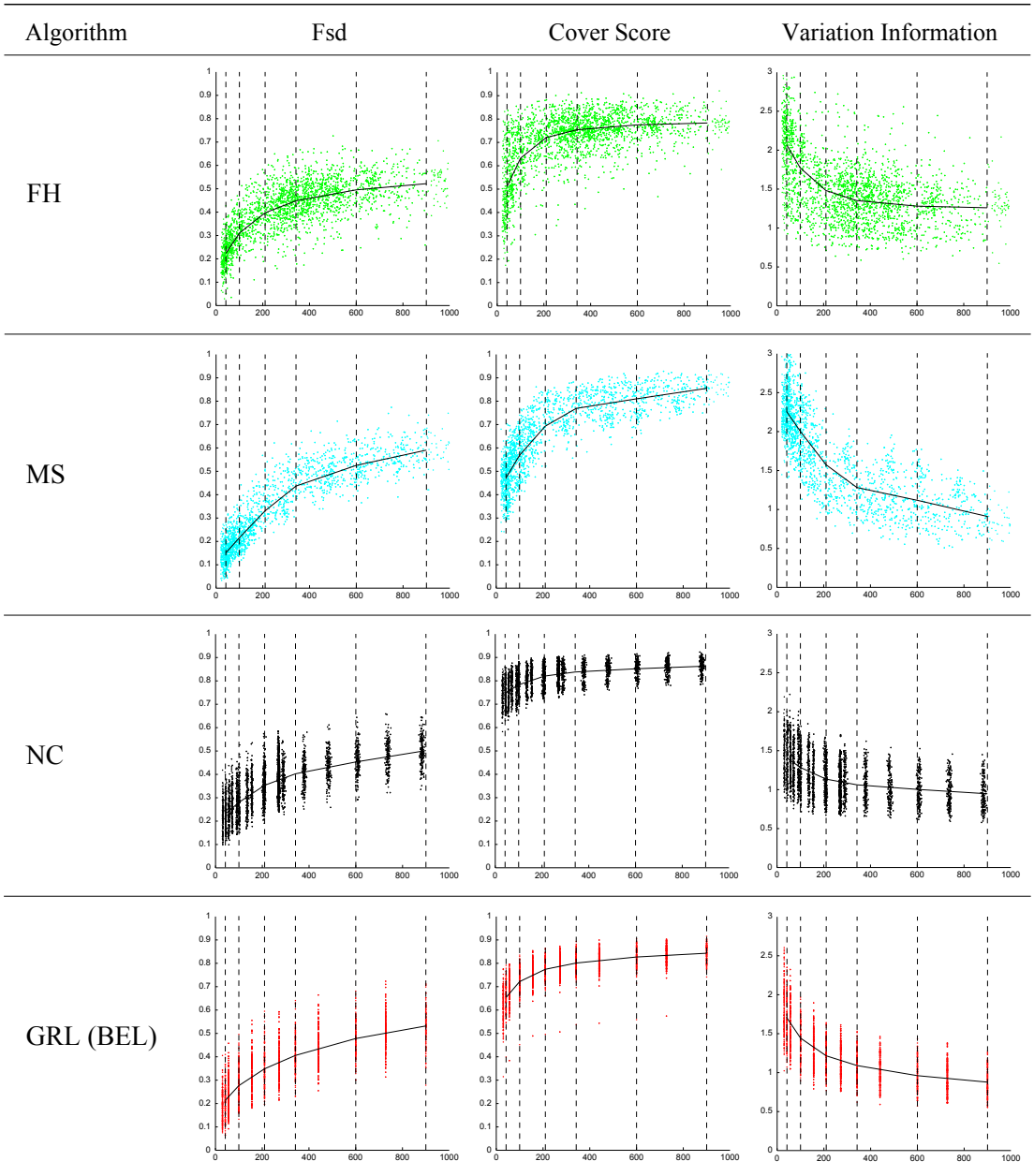
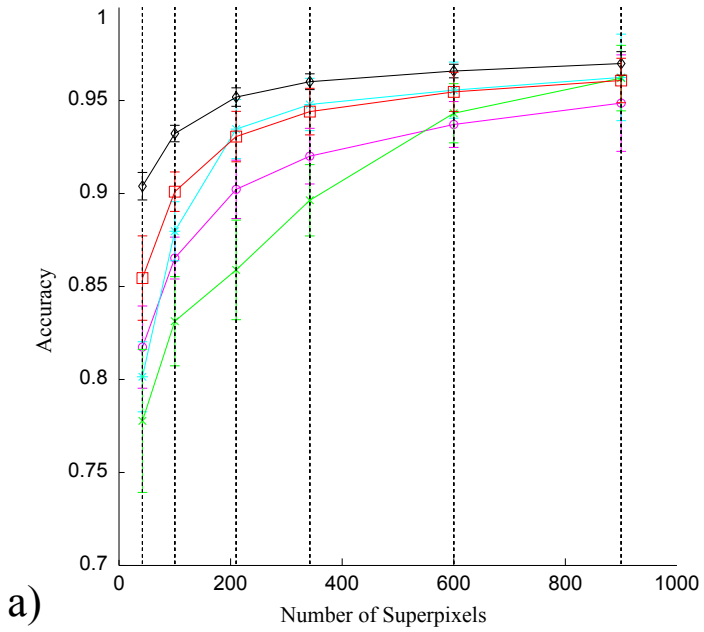


Figure 3.19: Algorithms vs Metrics on CamVid test data. Graph to show the distribution of data for the sets of parameters for each algorithm. Plots display the number of superpixels vs chosen metric of performance. Dashed black lines indicate interpolated results for six superpixel resolutions.

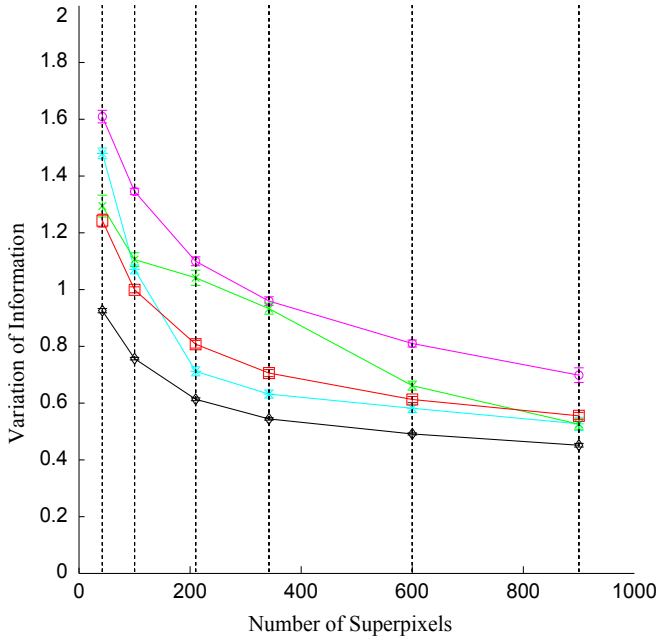
The ranking of the algorithms based on the Borda score for the three datasets can be seen in Table 3.5. In summary, the NC algorithm performs best across all datasets based on the combined quality measures. However our algorithm, GRL comes second overall across the combined datasets which is a notable result given the added topological constraint. Our method is much faster than the NC algorithm and we discuss this in more detail in Section 3.6. Moreover, it can outperform the MS and FH algorithms that are commonly used as superpixel pre-processing steps in vision pipelines. MS outperforms FH which is a result that is similar to others found in the literature [172, 264]. Full results for all measures are given in Appendix B. Comparative examples for low-mid-high ranked images can be seen for each dataset in Figures 3.21-3.26.

We make several observations about performance of the different datasets. Figure 3.20a shows the results for the Accuracy vs number of superpixel on the BSD. The hull is dominated by the NC algorithm across all resolutions. GRL performance is very similar to MS with slightly better performance at lower resolutions. Performance differences based on standard errors are indistinguishable at high resolutions and interestingly at 900 superpixels uniform sampling produces comparable results to other superpixel algorithms. While it is worth noticing that uniform sampling can produce high results it also suggests that simple Accuracy is not discriminative enough to capture all the properties of a useful segmentation. Alternatively, consider Figure 3.20b where we seek a minimum of the measure of Variation of Information vs number of superpixels. Here a similar pattern applies where NC dominates the hull but this time at high resolutions there remains a noticeable difference between uniform sampling and competing methods. This highlights the need to use several different measures to capture performance differences.

Our method is ranked third overall on the BSD but performance is similar to MS. In particular MS does particularly well on the F_{sd} measure compared to GRL and NC which produce very similar performance except at the lowest resolution. FH seems to do rather poorly using the Accuracy measure where even uniform sampling produces better results at lower resolutions but the overall rank shows its performance is in general much better. The FH algorithm lacks any compactness constraints which means at low resolutions it tends to merge or 'bleed' between large regions. For example, if we compare mid ranked images in Figure 3.21 the FH algorithm with 71 superpixels



a)



b)

KEY: Uniform, MS [53], FH [88] NC [185], GRL (BEL)

Figure 3.20: Selected performance measures on BSD. NC dominates the hull on both performance measures and GRL and MS have similar performance. Note that at the highest resolution of uniform sampling the algorithm performs similarly to competing methods.

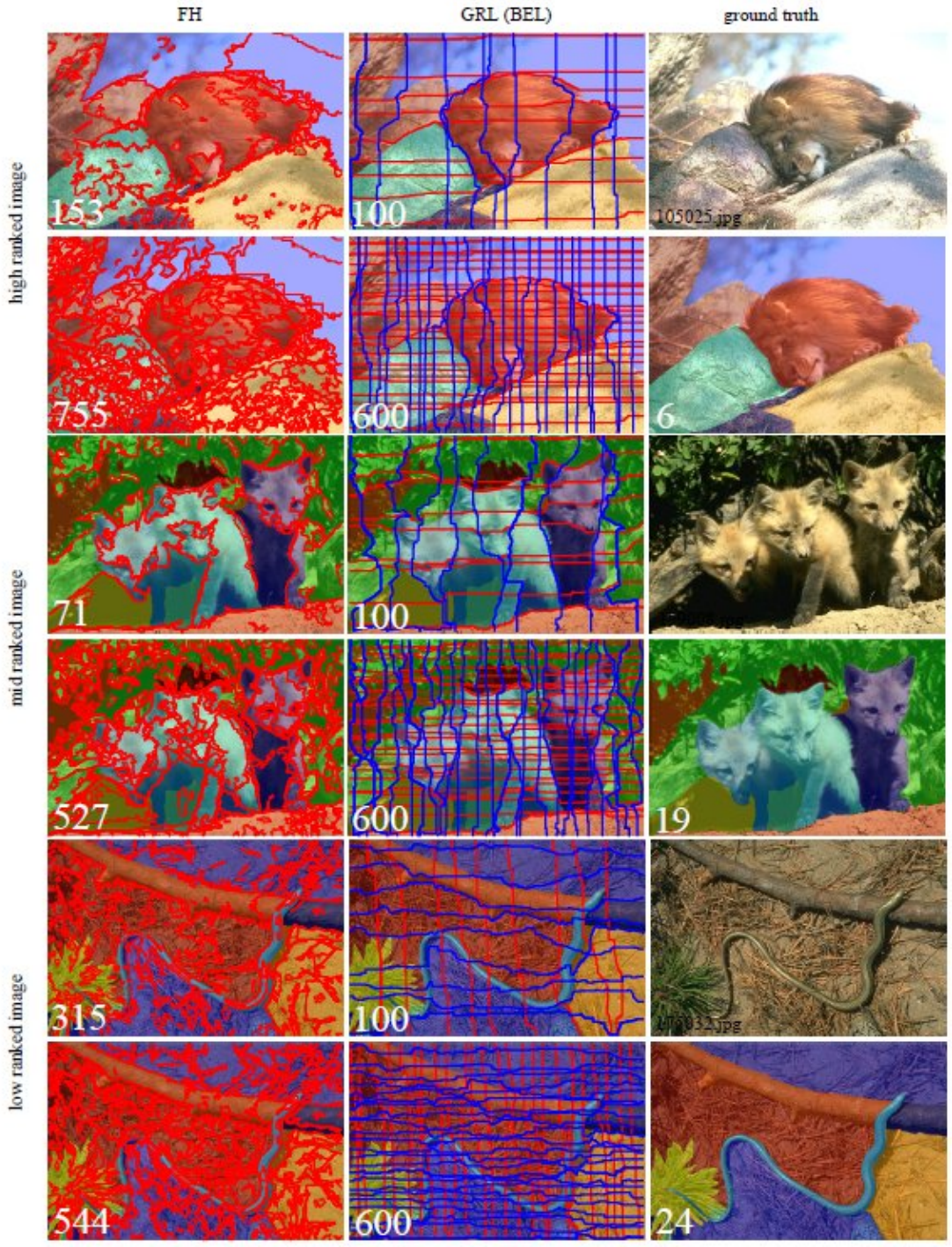


Figure 3.21: High/Mid/Low ranked images on BSD for FH and GRL algorithms. The last column shows the original image and human labeled ground truth. In general images with few ground truth regions produce high performance. Low ranked images tend to be images where the object and background share similar colour distributions. For images with weak boundary maps our method just produces slightly meandering paths that produce a regular grid over the whole image.

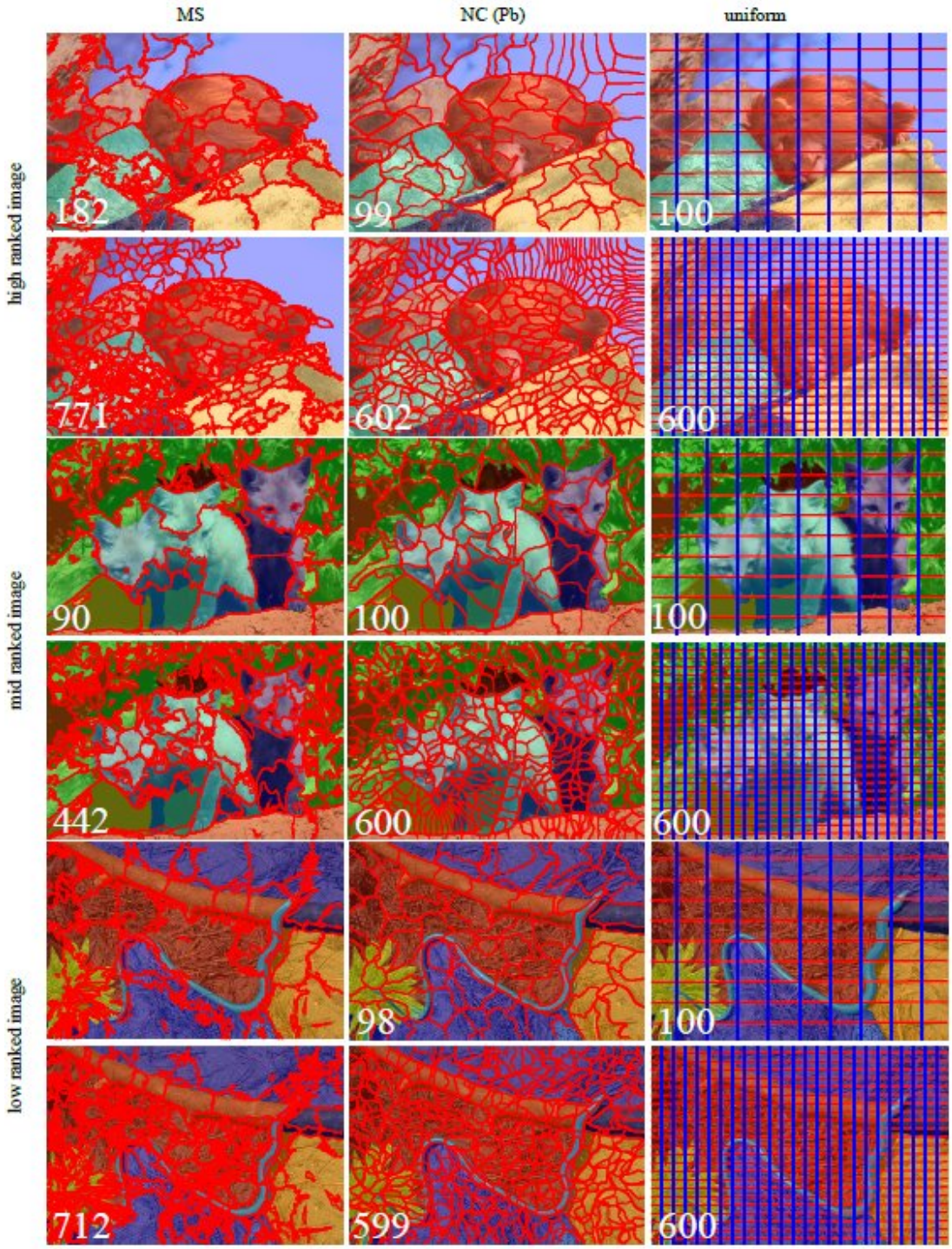


Figure 3.22: High/Mid/Low ranked images from BSD for MS, NC and Uniform algorithms. Compare images with Figure 3.21. In general uniform sampling is less able to capture the ground truth at low resolutions because the piece-wise linear approximation becomes a poor representation of object boundaries.

merges the head of fox on the left hand side with the background. In contrast our GRL algorithm with 100 superpixels separates these two ground truth regions. This is not a like-for-like comparison but illustrates the benefit of the regularizing constraints of our method. Modes of failure for our method are presented in Section 3.9. Figures 3.21 and 3.22 give comparative examples of high/mid/low ranked images on the BSD dataset.

Results for the VOC dataset can be seen in Table 3.5 column four. It was necessary to reduce the size of some images when using the NC algorithm to limit the memory requirements (reducing images of dimension greater than $x \times y = 640000$ pixels by 75%). This is likely to lower performance on some of the largest images. Our method is ranked second equal, indistinguishable from the MS algorithm using the Borda score. Figures 3.23 and 3.24 give comparative examples of high/mid/low ranked images on the VOC dataset.

On the CamVid data our method is ranked second overall. Again it was necessary to run the NC algorithm on reduced sized images (50%) because of memory requirements but this does not seem to unduly limit performance when compared to other algorithms.

Uniform sampling performs well against the MS and FH algorithms from which we draw two conclusions. First, the two algorithms that perform well (GRL and NC) use boundary maps, that contain learned information about the likely occurrence of object boundaries, whereas MS and FH essentially use local colour information. This suggests that more *a priori* information is required for good performance on the CamVid dataset. Secondly, it suggests that in the absence of strong object/background information, uniformly sampling the image produces good results - which we attribute to its conservative nature. This is interesting because the *null* segmentation [110] for our method, ie. a segmentation in the absence of any information in the local signal, is a uniform grid. Figures 3.25 and 3.26 give comparative examples of high/mid/low ranked images on the VOC dataset.

We conclude that our algorithm produces useful segmentation performance and is not unduly limited by the topological constraint. It is also instructive to consider the absolute numbers in the tables in Appendix B. If we look at average absolute performance across algorithms and resolutions for positive performance measures (F_{sd}, CoverScore, Rand Index and Accuracy) we get scores of 0.834, 0.827, and 0.732 for the

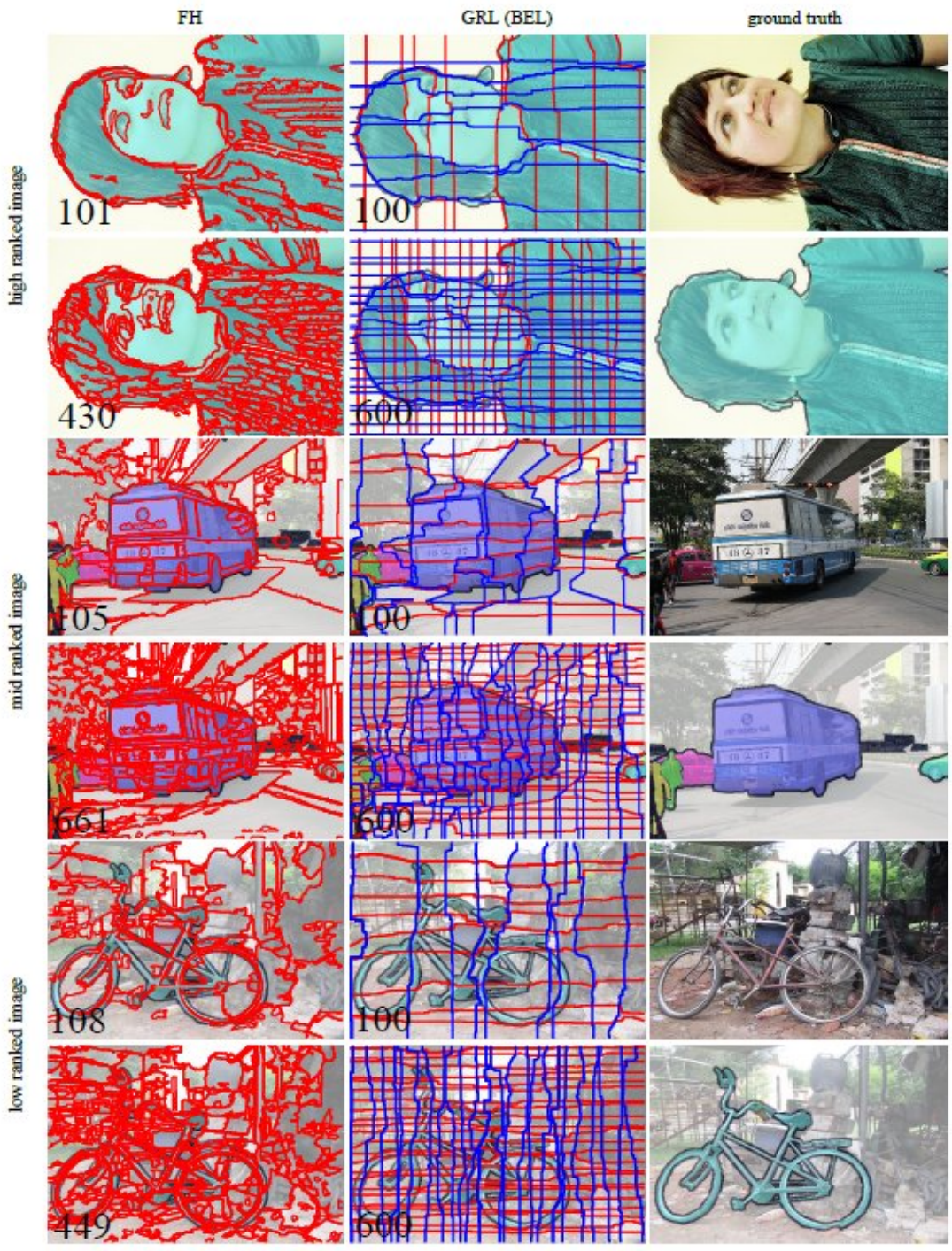


Figure 3.23: High/Mid/Low ranked images from VOC for FH and GRL algorithms. The last column shows the original image and human labeled ground truth. In general images with few ground truth regions produce high performance. Low ranked images tend to be images where the object and background share similar colour distributions. For images with weak boundary maps our method just produces slightly meandering paths that produce a regular grid over the whole image.

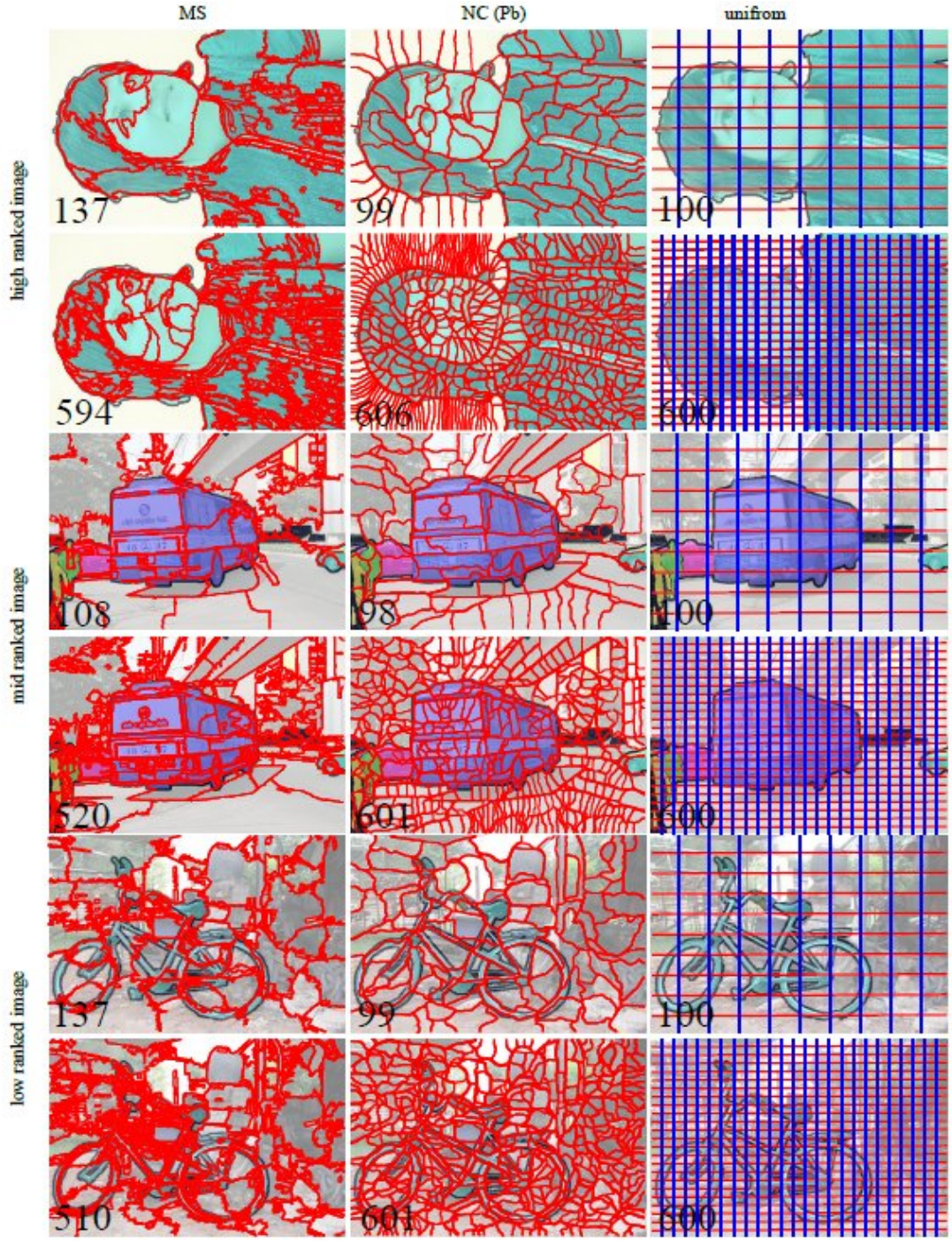


Figure 3.24: High/Mid/Low ranked images from VOC for MS, NC and Uniform algorithms. Compare images with Figure 3.23. In general uniform sampling is less able to capture the ground truth at low resolutions because the piece-wise linear approximation becomes a poor representation of object boundaries.

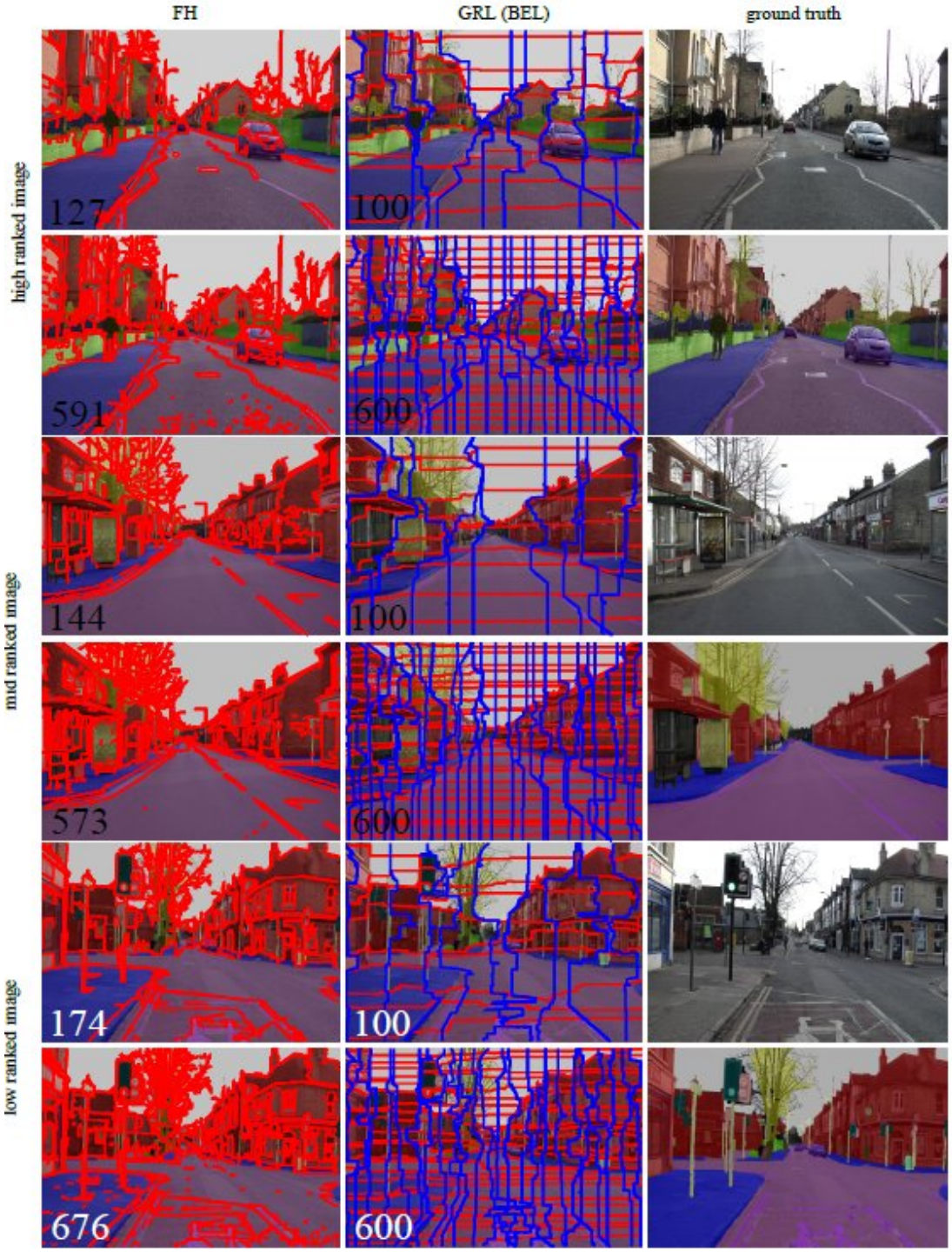


Figure 3.25: High/Mid/Low ranked images from CamVid for FH and GRL algorithms. The last column shows the original image and human labeled ground truth. In general images with few ground truth regions - pedestrians and vehicles on the road - produce high performance. Low ranked images tend to be images where the object and background share similar colour distributions or difficult classes - eg. column/pole.

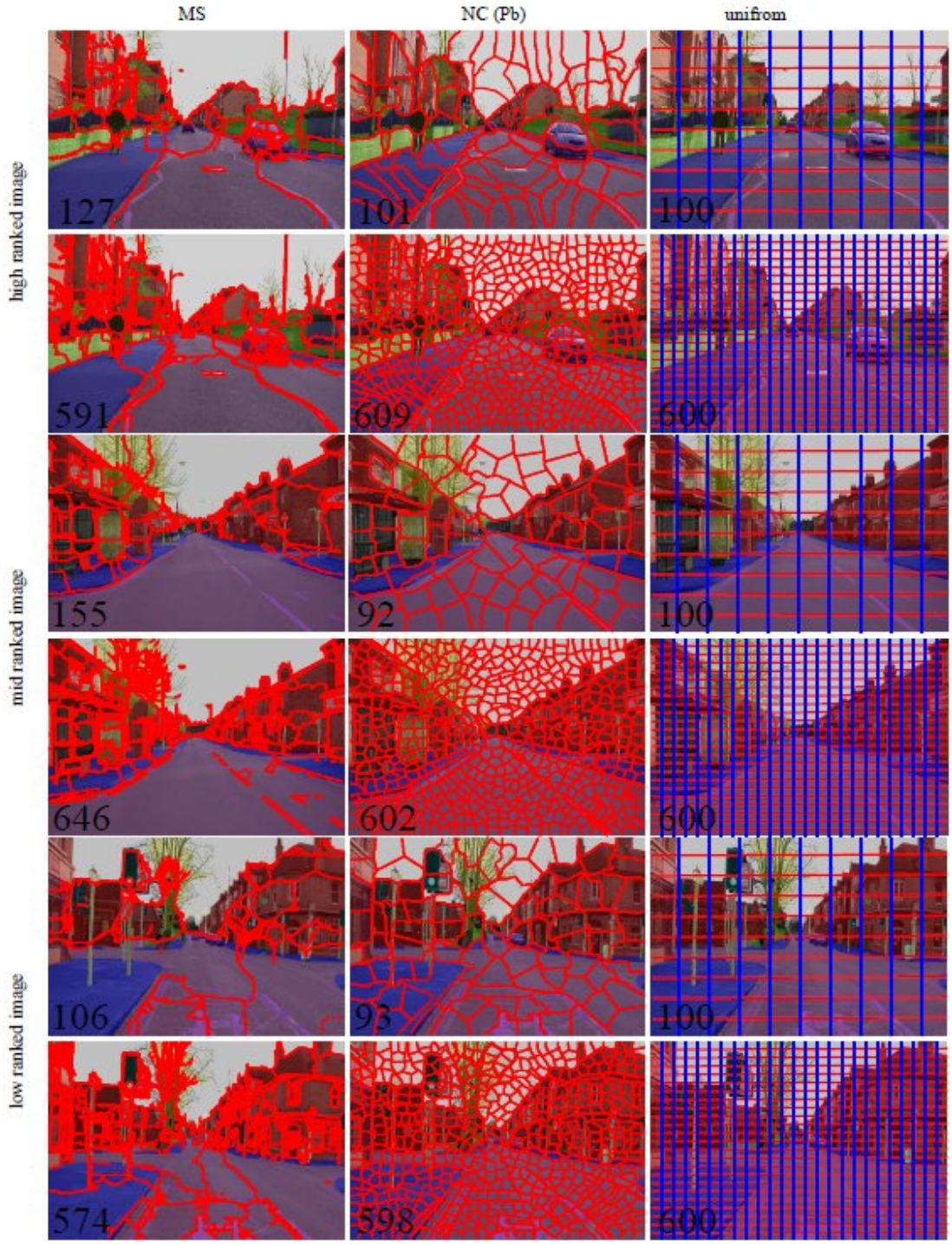


Figure 3.26: High/Mid/Low ranked images from CamVid for MS, NC and Uniform algorithms. Compare images with Figure 3.25. In general uniform sampling is less able to capture the ground truth at low resolutions because the piece-wise linear approximation becomes a poor representation of object boundaries - particularly in scenes with dominant 45° lines.

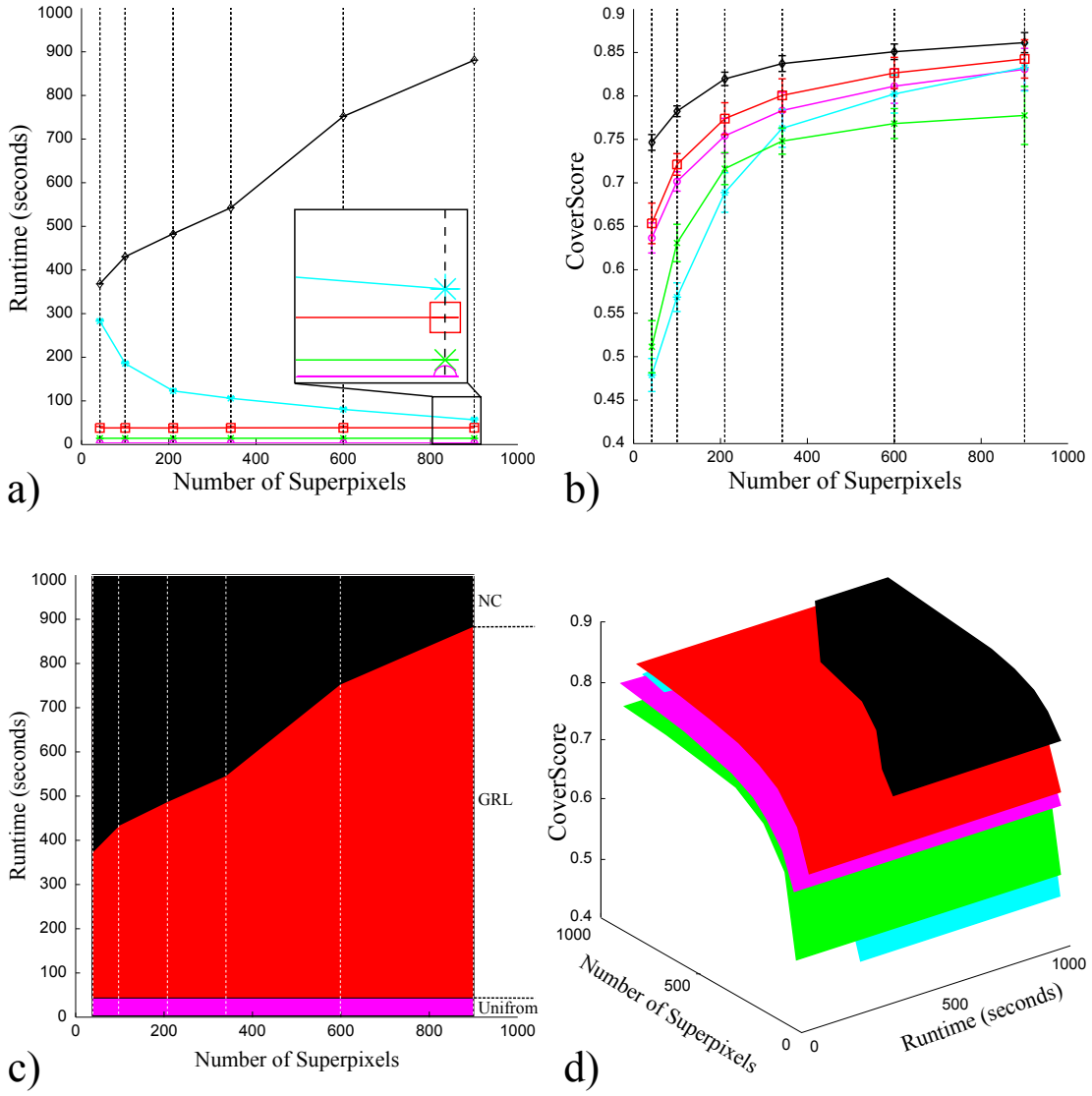
BSD, VOC and CamVid datasets respectively. We can see that performance is $\sim 10\%$ lower on the CamVid dataset and this is investigated further in Chapter 4.

3.6 Runtime and Computational Complexity

In the last section we concluded that the NC algorithm produces the best performance across datasets. However, the performance of our algorithm is encouraging, particularly when we consider the runtime of the algorithms. Figure 3.27a shows the runtime for competing methods on the CamVid images against the number of superpixels. This includes the time taken to compute the boundary maps in the case of the NC and GRL algorithms. Our timing evaluations are based on an Intel(R) Xeon(R) CPU 2.49GHz with 4GB of RAM.

Where processing time becomes a factor in an algorithm's evaluation NC no longer dominates the hull in the fitness/cost space. To demonstrate this let us use the Cover Score measure (Figure 3.27b) and consider the three dimensional space of $CoverScore \times Numberofsuperpixel \times Runtime$. A 2D projection of this space (with the Cover Score axis going into the page) can be seen in Figure 3.27c. The hull is now shared fairly evenly between GRL in the low runtime region and NC in the high runtime region. In this space uniform sampling also appears in the hull in the very low runtime region [84]. This explicitly shows the trade off between runtime and performance quality when evaluating superpixel algorithms.

Furthermore, if we discount the processing time for calculating the boundary map then our method becomes faster than the FH method, which is the second fastest algorithm in Figure 3.27a. Our partitioning method runs in an average 1.65s on the 720×960 CamVid images compared to an average 11.0s seconds for the FH algorithm. Discounting the cost of computing the boundary map may be valid if it is required for subsequent processing steps ie. the cost of calculating it is not solely for the segmentation stage, which might be the case for some applications. In practice the BEL boundary map is based on a boosted classifier and algorithms with similar structures have been shown to work at frame rate [240] so in many applications that computation of the boundary map is unlikely to be the limiting factor. While the analysis of the runtime is instructive it is implementation dependent. It is therefore useful to examine the complexity of competing methods.



KEY: Uniform , MS [53], FH [88] NC [185], GRL (BEL)

Figure 3.27: Comparison of algorithm runtime. a) Runtimes for competing algorithms on the CamVid dataset including both the time taken to compute the boundary map and the time for segmentation. b) Cover score vs number of superpixels on CamVid dataset. c) A 2D projection of $CoverScore \times Numberofsuperpixel \times Runtime$ space with the Cover Score axis going into the page. Note that in this combined cost space the hull is now not dominated by NC but shared fairly evenly with GRL. GRL dominates at lower runtimes. d) Full 3D projection of $CoverScore \times Numberofsuperpixel \times Runtime$ space.

The most popular normalized cut formulation proposed by Shi and Malik [238] uses a spectral approximation method with (approximate) complexity of $\mathcal{O}(N^{3/2})$ [159], where N is the number of pixels. The approximation involves solving an eigenproblem using k iterations of the Lanczos method and the space and run-time complexity also depend on the number of required superpixels. The method takes the order of minutes to converge per image and becomes prohibitive with large numbers of superpixels. In practice we have to resort to reduced size images and a limited number of superpixels. In these experiments we have made use of the implementation made available by [186] which takes the initial segments from the eigensolver and post-processes them using K-means. It is possible to make several further approximations to reduce runtime. Sharon et al. [235] reduce the complexity by a factor of $\sqrt{N}$ using a recursive coarsening scheme and Cour et al. [59] produce a linear time algorithm by solving a restricted problem. However, in practice we have found that this method remains slow, in the order of minutes for large numbers of superpixels, and memory requirements are prohibitive for large images. We have therefore persisted with the implementation of [186] in these experiments, reducing image size where required.

Predicting the runtime of `MS` is not straight forward as convergence depends on the properties of the feature space [46]. If we have N datapoints in D dimensions (eg. $D = 5$ for color images with spatial co-ordinates) then the average number of iterations per data point is k and a simple implementation has complexity of $\mathcal{O}(kN^2D)$. In practice several techniques are used to speed up the EDISON implementation including not applying the Mean Shift procedure to pixels which are on an existing trajectory. Despite this, runtime remains dependent on the kernel sizes and the method is slower than our method at all superpixel scales, even including computing the boundary map. However, real-time implementations of `MS` have been reported [201] and while we are unable to comment on the performance changes the approximations may introduce the absolute speed of the EDISON implementation should not necessarily be seen as a limiting factor.

The computation for `FH` [88] based on Kruskal's algorithm for the minimum spanning tree is dominated by the need to sort the edge weights. This usually yields a run time of $O(N \ln N)$ complexity on grid graphs.

Our method is also fast and approximately linear as the number of superpixels

increases. If we ignore the strip overlap then for an image with $\sqrt{N} \times \sqrt{N}$ pixels there are approximately $2\sqrt{N}/S$ strips of length $\sqrt{N}$ and width S. We saw in Section 3.2.4 that the partitioning part of our method is dominated by the need to maintain a priority queue used in Dijkstra's algorithm with complexity $\mathcal{O}(V \ln V + E)$. This gives an approximate complexity of $\mathcal{O}(S\sqrt{N} \ln(S\sqrt{N}) + 4S\sqrt{N})$ which is in practice faster than the `FH` algorithm for the range of superpixels we have investigated. It is possible to implement different data structures that can solve the single-source shortest path problem in linear time [255] but we have not explored use of them in this work.

We conclude that our method is very efficient which makes it useful as a pre-processing step in vision pipelines. Moreover, the separation of the image into strips means that it is relatively simple to parallelize on overlapping strips.

3.7 Connectivity

A non-regular topology is undesirable for the reasons highlighted in Figure 3.13 and discussed in Section 3.4. Additionally, it also has an unpredictable effect on the mean number of neighbors of each superpixel. This is important as message passing algorithms [204], commonly applied to random field models, pass information between adjacent nodes. To evaluate the connectivity produced for each algorithm we construct the region adjacency graph (RAG) for each superpixel lattice on the BSD. The RAG represents each superpixel as a node in the graph with edges between superpixel which share a common boundary. Example RAGs can be seen in Figure 3.28.

Figure 3.29 shows the mean connectivity for a node at different resolutions of superpixels for competing methods on the BSD. We can see at the lowest resolutions the `FH` algorithm has the lowest connectivity. However as the resolution increases the mean number of connections per node increases for competing methods. While competing algorithms have fewer neighboring superpixels than a regular 8 connected lattice there is no principled way of reducing connectivity without further processing. In contrast it is simple to impose 4 connectivity in our regular lattice which may be important if the cost of operations between nodes is high for a given application.

More importantly, lattice connectivity itself can bring additional advantages. For instance, a grid graph is bipartite. In other words, we can divide the graph into two sets of nodes that are only connected to nodes in the other set. The bipartite property can

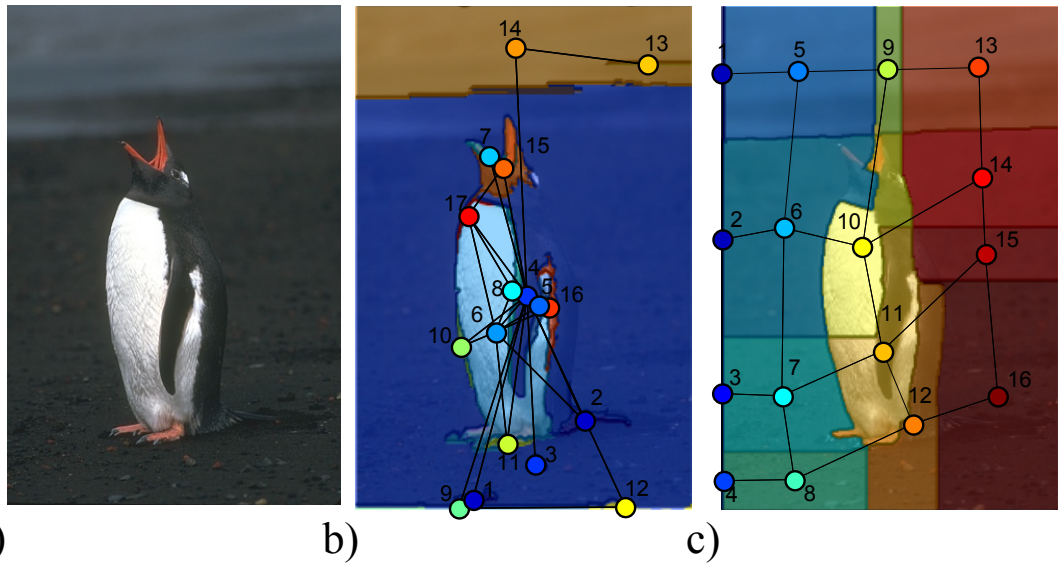
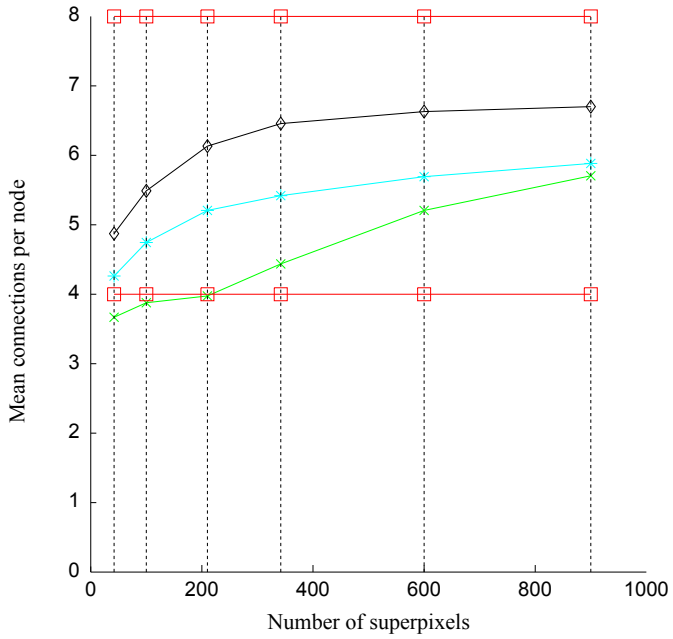


Figure 3.28: Region adjacency graphs. a) Original image. (see Figure 3.2) b) Image graph produced using FH algorithm [88]. Note the irregular connectivity. c) Image graph produced from our GRL algorithm. The grid provides consistent connectivity and a more uniform distribution of superpixels. The grid graph conveys additional advantages, see Section 3.7 for details.



KEY: MS [53], FH [88] NC [185], GRL (BEL)

Figure 3.29: Connectivity. Graph to show the mean connectivity of regions vs number of superpixels on the BSD. All competing methods increase mean connectivity as the number of superpixels increases. Both 8 and 4-connected regular grids are shown for our lattice algorithm.

be exploited to reduce memory requirements of message passing algorithms [91]. The symmetry of a grid also can be exploited by alternately processing horizontal and vertical directions in certain graph representations [206, 229]. Perhaps more significantly, some labelling problems become more tractable on grid graphs. For instance, Carr and Hartley [44] present an algorithm based on 4-connected grids that can be used to minimize non-submodular binary problems. Recently, Schraudolph [233] also demonstrated solving non-submodular binary MRFs with a technique that leverages a graph embedding based on a regular grid.

Lastly, we shall see in the following section that it is possible to apply additional algorithms to our superpixel lattice. For instance, we apply the `MS` algorithm which can also exploit heuristic speed ups based on grid structure [47].

We conclude that the regular lattice not only imparts engineering benefits but that it is also possible to exploit fixed connectivity algorithmically.

3.8 Merging Superpixels

The conclusion of Section 3.5.4 was that the performance of our algorithm is encouraging, and this conclusion is re-enforced when we consider the runtime of the algorithm. However, it is instructive to try and understand what limitation the topology constraint has on its performance. One way to observe this is to use the ground truth boundaries as the boundary cost map to see whether it was possible to achieve good performance using the greedy lattice. Table 3.6 shows ranked results for competing algorithms, where we have used human labeled ground truth (GT) data to produce a binary boundary map as input for our lattice algorithm.

Unsurprisingly, the segmentation using human ground truth dominates performance scores compared to the those generated using algorithmic boundary map methods. However, it is more instructive to examine absolute scores. For instance, if we consider Cover Score on the the VOC dataset our method produces a maximum performance of ~ 0.93 (at resolution of 900 superpixels) which is well above other segmentations (max ~ 0.86) but still significantly less than the maximum possible score of 1. If we were to run competing algorithms whilst allowing them to exploit ground truth data then they would likely achieve close to maximum performance. We conclude that our greedy method for maintaining the lattice structure does decrease performance, but

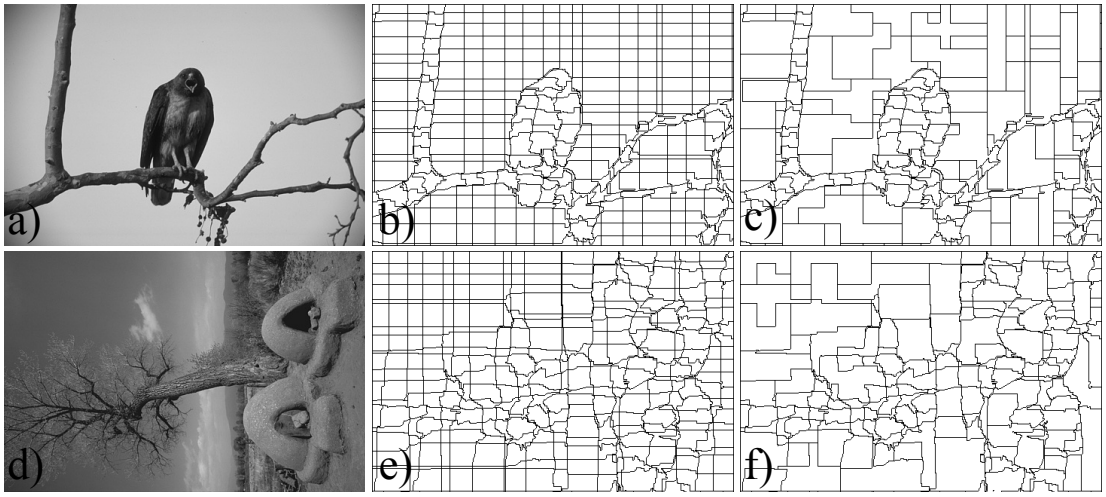


Figure 3.30: Merged lattices using [74]. This reduces the number of superpixels while the lattice preserves structure as well as accuracy. a) Original high ranked image (easy). b) 20×20 regular lattice. c) Merged lattice reduced to 200 superpixels. Accuracy reduces from 0.956 to 0.951. d) Original low ranked image (difficult). e) 20×20 regular lattice. f) Merged lattice reduced to 200 superpixels. Accuracy reduces from 0.923 to 0.902.

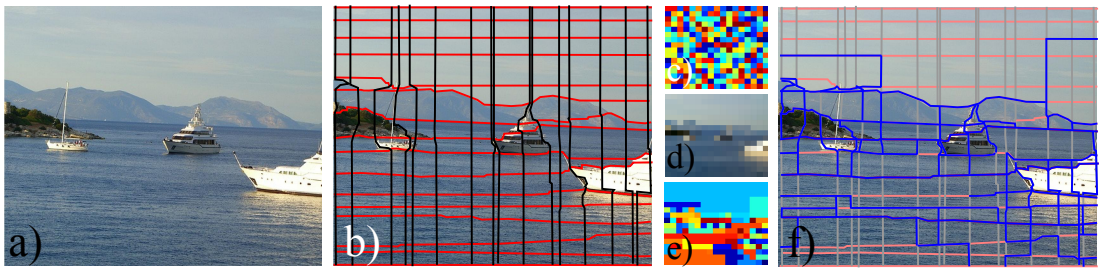


Figure 3.31: Merging lattices with the Mean Shift algorithm. a) Original VOC image b) GRL using BEL boundary map. c) Randomly colored image with same number of pixels as there are superpixel in the lattice. d) Mean image based on superpixels. e) Low resolution image segmented using the Mean Shift algorithm. f) Original GRL with boundaries that are preserved after subsequent merging shown in blue.

Algorithm	Reference	Datasets			
		BSD [173]	**VOC** [87]	**CamVid** [39]	Total
GRL (BEL)	[182] ([74])	55	70	74	199
GRL (GT)	[182]	**108**	**108**	**108**	**324**
NC (Pb)	[186] ([174])	94	81	92	267

Table 3.6: Algorithm ranking using Borda score on three datasets and combined totals. Unsurprisingly, the segmentation based on the boundary map generated using human labeled ground truth (GT) dominates performance scores. However, it is more instructive to examine absolute scores, see text for details.

that this decrease in performance is not currently the limiting factor in producing useful superpixels.

However, we know that this limitation reduces asymptotically as the the number of superpixels reaches that of pixels, because the ground truth maps directly onto the grid graph of pixels. There is therefore a trade off between the lattice structure at low resolutions and its ability to capture the structure of the ground truth data.

One way to highlight this is to note that our method necessarily divides homogeneous regions into multiple superpixels. Whilst this is an important property of our algorithm it also limits performance on certain images because competing algorithms can concentrate the modeling power of their superpixels in regions of the image that require more detail.

A possible way to alleviate this is to take a fixed high resolution lattice and then subsequently merge superpixels on this lattice down to a lower resolution. In other words, we can apply a further segmentation algorithm to our original superpixel lattice. By doing so we attempt to exploit the increased detail captured at the higher resolution lattice but reduce the number of superpixels to more accurately account for the detail in a particular image. Moreover, this merging process does not eliminate the desirable properties of the lattice structure. For instance, in the context of a message passing algorithm, we consider the merged regions as groups of constituent superpixels: they maintain their usual relationships with neighbors outside the group, but the MRF/CRF costs are designed so an infinite penalty is incurred if the group members take different values. Many current inference algorithms can operate in these circumstances.

In a previous publication of this work [182] we demonstrated improved performance of subsequent merging on a high resolution lattice by greedily merging superpixels based on the edge strength of their boundaries on the original boundary map, ie. removing boundaries with the smallest cost. An example of this along with some results are shown in Figure 3.30.

Here we demonstrate the versatility of having a superpixel algorithm that conforms to a grid by applying the MS algorithm directly to the superpixel lattice. This process is illustrated in Figure 3.31. We begin with an image and compute a high resolution lattice, see Figure 3.31b. This lattice of superpixels can be represented as a low resolution grid graph similar to the original pixel image, Figure 3.31c, and we can create a low resolution image by averaging the values of pixels within each superpixel, Figure 3.31d. It is then possible to run standard image algorithms on this small image, eg.additional segmentation algorithms, Figure 3.31e, which allows us to generate an image with a reduced number of superpixels.

The relationships between the new regions are inexpensive to calculate as we operate on low resolution images, tens or hundreds of pixels, rather than hundreds of thousands of pixels of the original images. The additional time taken to apply MS on a superpixel images is a couple of thousandths of a second which is negligible compared to the original segmentation algorithms. Additionally, our algorithm runtime is practically linear in the number of superpixels (for the range of superpixels we have investigated see Figure 3.27a) so starting with a higher dimensional lattice does not change our runtime performance.

The effect that this additional merging has on performance can be seen in Table 3.7. Relaxing the topological constraint in this way improves performance. Our method is now ranked first in two of the datasets and second in a third, despite a fast runtime and the of the lattice to a regular grid.

3.9 Failure Modes

During the course of this chapter we have demonstrated that the GRL algorithm produces good performance when compared to competing methods. There are hard examples that all algorithms find difficult. Example images include heavily camouflaged objects or confusing cases with reflections and occlusions. However, given that our

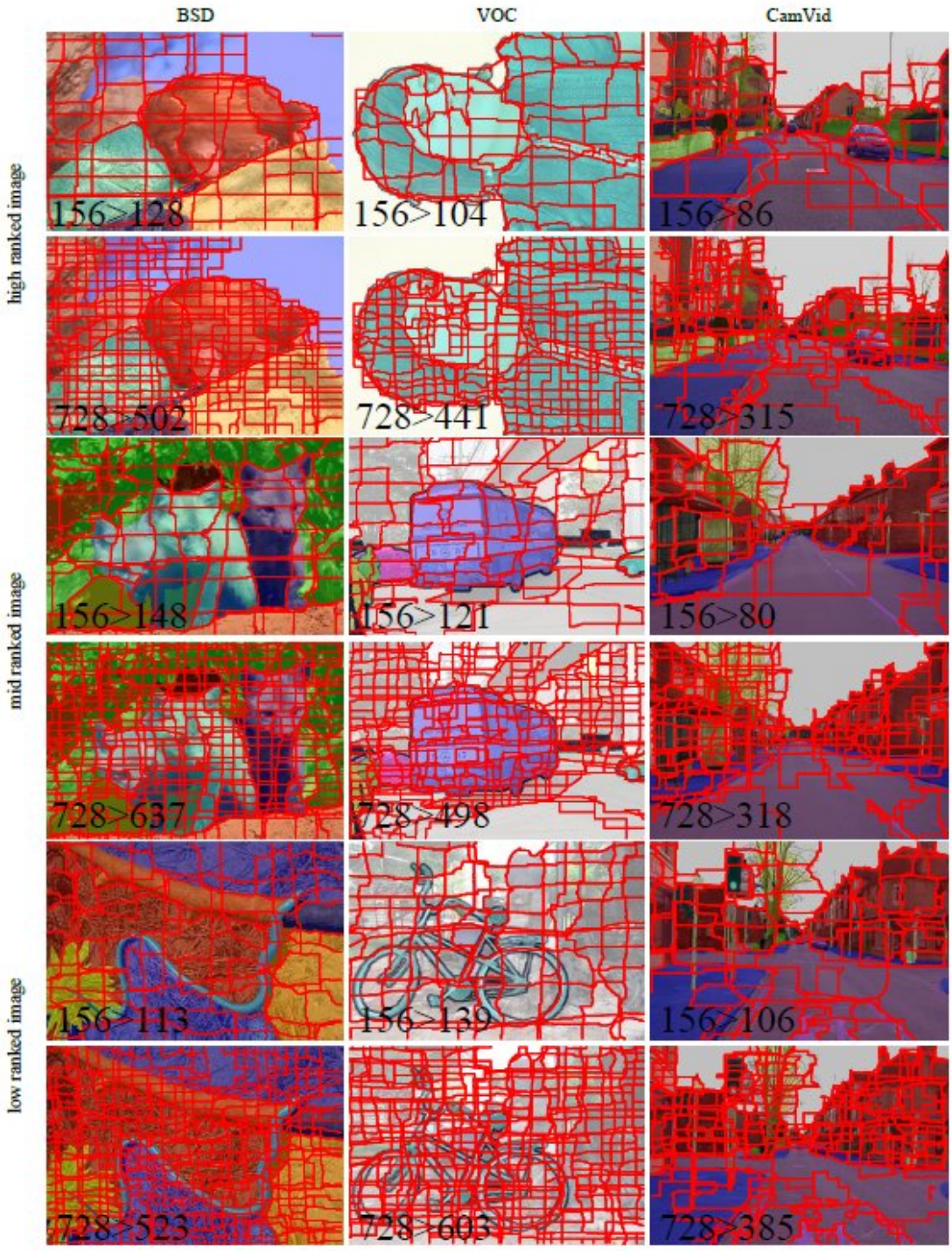


Figure 3.32: High/Mid/Low ranked images from BSD/VOC/CamVid datasets. Compare with segmentations in Figures 3.21-3.26. Underlying low resolution grid is preserved but adjacent superpixels can be merged to produce fewer superpixels.

Algorithm	Reference	Datasets			
		BSD [173]	**VOC** [87]	**CamVid** [39]	Total
GRL (BEL)	[182] ([74])	104	120	117	343
GRL-[MS] (BEL)	[182] [53] ([74])	112	**133**	**131**	378
MS	([53])	112	124	85	321
NC (Pb)	[186] ([174])	**136**	132	130	**398**

Table 3.7: Algorithm ranking using Borda score on three datasets and combined totals. Improved performance when merging superpixels can be seen in all datasets. In two datasets the GRL algorithm produces state-of-the-art performance even with the underlying topological constraint. Maximum Borda score for individual dataset = 144.

algorithm has some novel additional properties and restrictions it is instructive to look at where and how it fails in more detail. Figure 3.40 illustrates examples of selected failure modes of our algorithm using image pairs. The left image of the pair gives a schematic of the failure mode with horizontal superpixel boundaries shown in red and vertical superpixel boundaries in blue. The right image gives a real example of the failure on the BSD dataset. A general observation is that the position of superpixel boundaries are controlled by a path that is optimal over the full length of an image strip. This means that at a local level the boundaries can look poor if the path is responding to a feature in the boundary map that is a long way from its current position. Moreover, an indication of the problems of associated with a lattice is given by performance on the ground truth data. It is encouraging that it is not the limiting factor in performance, ie. with a perfect boundary map we get good results, but the fact that we do not achieve perfect results indicates that the lattice places restrictions on performance.

We discuss each failure in turn. In Figure 3.33 (Figure 3.40a) we can see a result where two orthogonal paths diverge before meeting on a boundary they are both currently following. This leaves a small region where there is strong boundary information on one side of a region that is not captured properly by either path. Because most of the boundary information is captured by both paths there is very little decrease in energy for conforming to the bound-

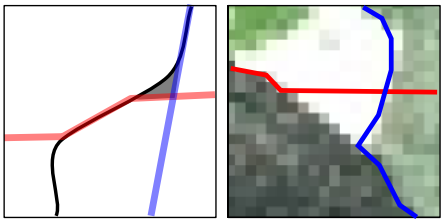


Figure 3.33: Cross-over error.

ary information in this local region and both paths are influenced by the boundary map elsewhere in the image.

Figure 3.34 (Figure 3.40b) illustrates an example where the path is influenced by boundary information on one side of and object/background region in such a way that it produces a "close off error" because although it may capture most of the boundary information it does not produce good object/background separation. The result is similar to "leakage" in agglomerative approaches, though the cause is different.


Figure 3.34: Close-off error.

Figure 3.35 (Figure 3.40c) illustrates a fairly common effect where the limited interaction between paths in orthogonal directions results in a background-object-tie effect (bo-tie - named not least because it commonly looks like a bow tie!). In this case a vertical boundary is mostly captured by two paths in the horizontal direction. This means there is very little remaining boundary information to influence paths in the vertical direction. Consequently, the central superpixel is not "closed off" and the object/background separation is poor.

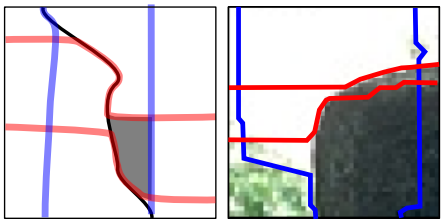


Figure 3.35: Bo-tie error.

In Figure 3.36 (Figure 3.40d) we see one of the major limitations of our greedy approach. The division of the image into overlapping strips means that the strip boundary can occasionally constrain the path of a superpixel boundary. In the real example (right) this produces a bo-tie effect where the central superpixel is assigned to object or background depending on the dominant class. This limitation is reduced by the contributions of Chapters 4 and 5 but an alternative approach would be to alter the boundary map for each orientation to reduce the effect of paths selecting boundaries in the orthogonal direction.

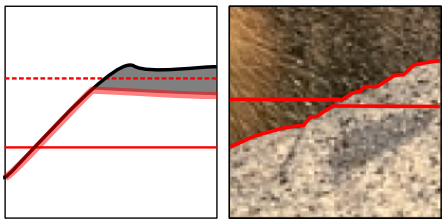


Figure 3.36: Strip error.

Understandably, the lattice has difficulty capturing image features that are particularly "un-lattice" like. One common feature that is badly represented are long parallel lines at an orientation that does not correspond to the lattice - ie. 45° - see Figure 3.37

(Figure 3.40e). It also produces a problem for interacting paths of different orientations as the real example demonstrates. The central red path approaches from the bottom left, starts on the bottom of the object, crosses to the top, and exits top right. These effects might be reduced by altering the boundary map to give greater preference to continuous boundaries.

Figure 3.38 (Figure 3.40f) illustrates A-like or Y-like junctions. These can cause a problem because the convergence of two superpixel boundaries is prevented in order to maintain a regular lattice. Often a small piece of object is incorrectly captured at the apex of the join between boundaries.

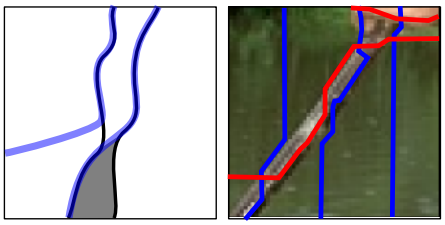


Figure 3.37: Diagonal error.

The tortuosity parameter is intended to encourage the final path to follow object boundaries but our shortest path formulation implicitly penalizes path length. This means that some tortuous features, see Figure 3.39 (Figure 3.40g), are poorly captured. In Figure 3.40h we can see that the separation into strips means that the filamentous objects are often only followed on one side of the boundary. Separation by simple superpixel boundaries also means that the separation of non-distinct or porous classes is poor, see in Figure 3.40i.

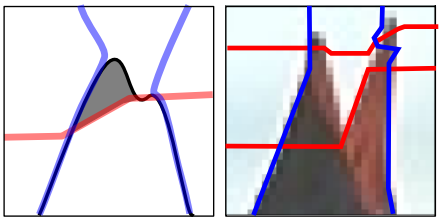


Figure 3.38: A/Y-like error.

Finally we can see in Figure 3.40j that object details below the resolution of the lattice are not captured well. In this example the lattice captures the edges of the building in the vertical direction and the 'window' feature is not well represented and this resolutions as it is merged with the surrounding wall to the left and right.

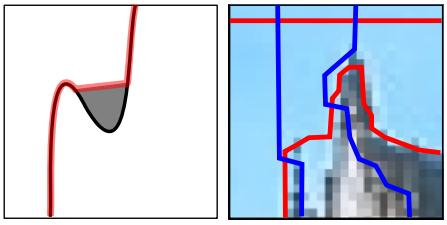


Figure 3.39: Tortuosity error.

In general the distribution of superpixels over the whole image means that small details that are captured in agglomerative methods are missed. In summary whilst overall performance is good our method does display failure modes that are particular to maintaining a lattice structure.

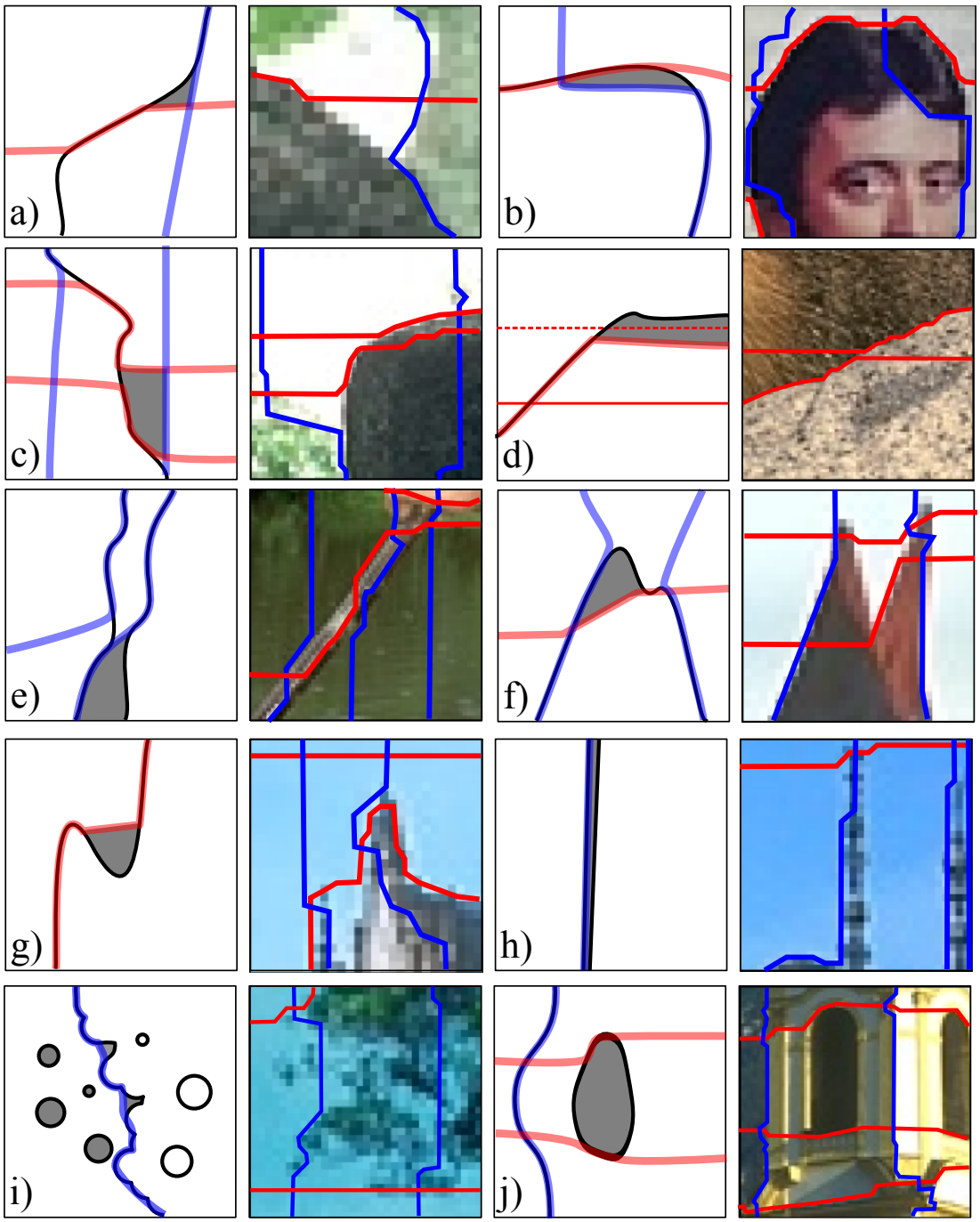


Figure 3.40: Failure Modes. Image pairs consisting of a schematic for error types (left) and a real example from the BSD dataset (right). For the schematic examples, object boundaries are black and poorly segmented regions colored gray. Vertical superpixel boundaries are blue and horizontal red.

3.10 Temporal Stability

In Section 3.3 we demonstrated stability across the parameter space of our algorithm. Another particularly important property is stability with respect to input image data - small changes to the input image result in small changes to the segmentation output.

We can extend the method presented in this chapter to deal with the temporal aspects of segmenting sequences of video footage. To do this we substitute the use of a shortest path algorithm used to find a minimum cost path within a strip (Section 3.2.4) to an st-MINCUT solution solution over several frames of an input sequence. In this new graph source and sink nodes are connected to the edge of each strip of the image over several frames rather than at each end of the strip. An example of the graph construction for this temporal method is illustrated in Figure 3.41a and the cut represents a surface across several frames of the image sequence 3.41b.

Solving the st-MINCUT problem in 3D means that there is a smooth transition from a superpixel in one frame to the next. However, this is only half a solution as either memory or time constraints will limit the number of available frames segmented at any given instance. Given such constraints, each batch (set of frames) would be temporally consistent but there would be a large jump at the transition between batches. However, it is possible to impose temporal stability on the next batch of frames by utilizing the greedy solution from the previous batch of frames.

One solution to achieve a stable greedy lattice is to alter the nodes and t-links in the 3D grid graph. The first frame of each new batch is connected (seeded) to the nodes in the graph of the last frame in the previous batch. Membership of nodes to source and sink in the s-t min-cut solution from this last frame are used to connect new t-links in the first frame of the new batch. An example of this can be seen in Figure 3.41a where the white and black nodes in the 'seed' frame effect the membership of nodes to source and sink in the subsequent batch of unsegmented frames.

To demonstrate performance we took 50 frames from a video sequence of a shop entrance in a shopping precinct which presents an empty scene with no moving objects. We segmented this sequence using `FH` and `GRL` algorithms. Example frames can be seen in Figure 3.42.

We quantify stability by comparing adjacent pairs of images in the sequence. As the `FH` algorithms does not produce segmentations that are isomorphic we find the

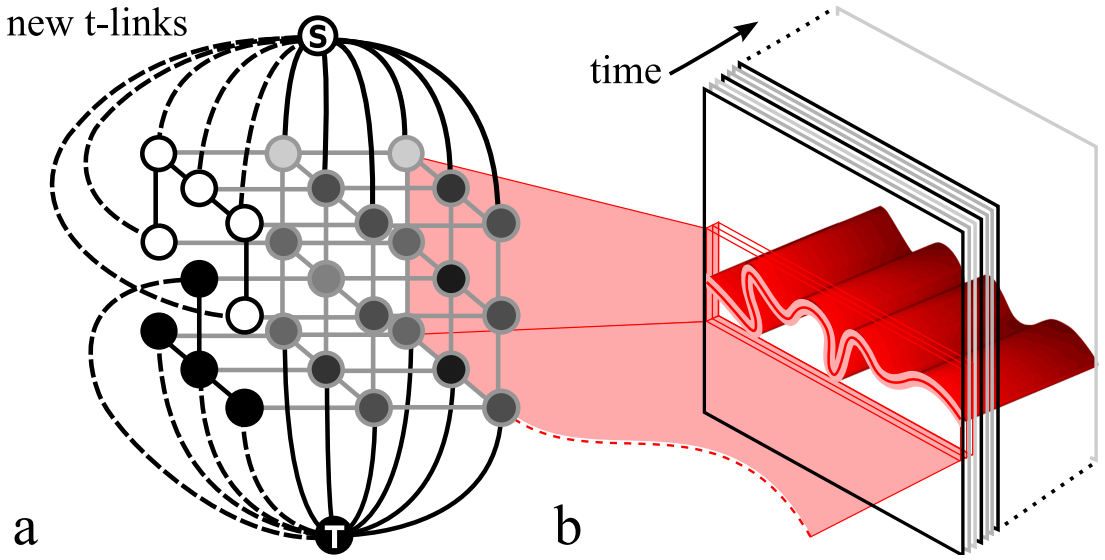


Figure 3.41: A seeded 3D lattice. a) A batch of two frames with additional t-link edges from the solution of previous frame (black outline). Additional nodes colored white and black from previous s-t min-cut solution. b) Cut surface representing the solution to 3D lattice. Performing the s-t min-cut over several frames results in a cut that is stable between consecutive frames.

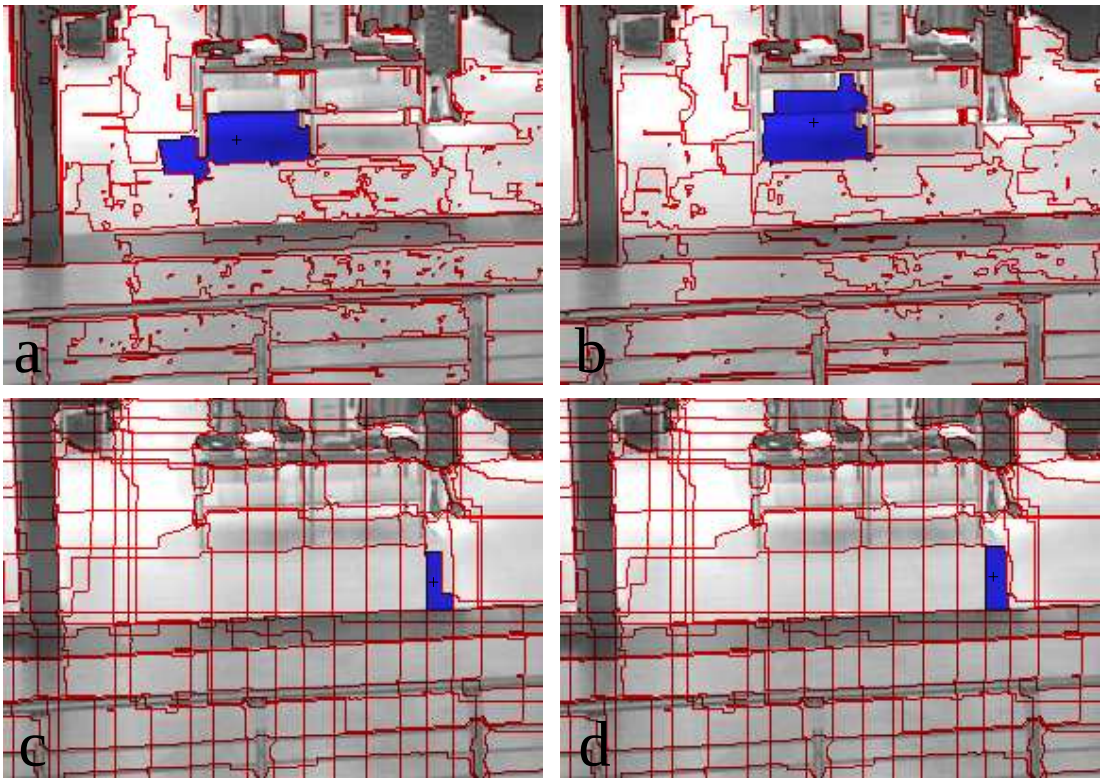


Figure 3.42: Superpixel Stability. Example superpixels (blue) have stability of 0.70 with the centroid marked with a cross (black). Note the change in superpixel boundaries (red) between consecutive frames. a-b) Example frames i and $i+1$ using [88], stability 0.69. c-d) Example frames i and $i+1$ using a greedy regular lattice across frames, stability 0.97.

nearest superpixel in the consecutive frames where proximity is determined by the Euclidean distance between superpixel centers. We then count the proportion of pixels from the superpixel in the first image that are in the matching superpixel in the second image. We normalize this by the total number of pixels to provide a number that varies from 0 (totally unstable) to 1 (completely stable). As the scene is static, viewed with a static camera, the ground truth stability is 1.

The FH algorithm produces a different number of superpixels in each frame, despite the near identical images. With the default settings the mean number of superpixels was 187, but the total number varied by 40 superpixels over the 50 frames.

The GRL algorithm applied to each frame produces a mean stability of 0.73 compared to 0.69 for the FH algorithm. However when using multiple frames and an st-MINCUT solution it rapidly increases to 0.96, 0.97 and 0.98 using batches of 1,2 and 5 frames respectively. Moreover, the failure mode of our algorithm differ to that of FH. For instance, if we compare the blue superpixels in 3.42 we can see superpixels that have the same stability. However, the centroid of the superpixel generated using our method differs only very slightly and the superpixel boundaries are the same on two sides whereas the shape and position of the closest matching superpixels in the FH display a greater degree of variation.

This simple experiment only addresses one possible, and simplest, scenario of *static-scene-static-camera*. There is a trade off between the stability or smoothness, and how accurately the boundary information is captured for each individual frame. This would be more apparent in dynamic scenes or changing camera viewpoint. For instance, other possible scenarios include *static-scene-moving-camera*, *moving-scene-static-camera* and *moving-camera-moving-scene*. Work on causal filtering (temporal stability) for the MS algorithm and other filtering algorithms is presented in [201] but comparing these methods is left for future work.

3.11 Discussion and Future Work

Although we have highlighted some notable failure modes of the algorithm presented in this chapter we have demonstrated that a superpixel algorithm that produces superpixels that conform to a grid does not unduly limit segmentation performance. If we relax the topological constraint, our method can out perform state-of-the-art superpixel

algorithms. We draw comparisons between our method and other techniques before setting out several possible extensions and improvements to the work in this chapter.

3.11.1 Relation to other work

The use of the dynamic programming algorithm for generating fast superpixels for video analysis was independently investigated in [77]. Their results are promising but their motivation is slightly different and they not use the path constraints to enforce the lattice topology on the final segmentation. Some of the advantages of having a grid structure, for instance the running time of message passing algorithms, were discussed in Section 3.7 and we demonstrated the versatility of a lattice by applying standard implementation of the `MS` algorithm to the superpixels generated by our method.

Another technique that relates to the work presented in this chapter is that of "Seam Carving" by Avidan and Shamir [20]. The difference between the two methods can be characterized by answers to an image retargeting problem: 1. Which pixels can we remove from a high resolution image? - ie. which pixels are redundant? or 2. Which pixels can we combine in a high resolution image? In this chapter we set out to answer the second phrasing of the problem and arrive at a segmentation algorithm. In contrast Avidan and Shamir [20] set out to answer the first phrasing of the problem but arrive at a very similar solution: they select pixels by finding minimum cost paths across an image.

Avidan and Shamir [20] investigate only the dynamic programming technique and do not impose the path constraints but it is interesting to consider the energy function they use. Their minimum cost paths are designed to find paths that avoid object boundaries and thereby reduce the possibility of creating new artifacts in the resized image. In our work we invert this energy so that the cost is low where object boundaries are present - and we therefore produce superpixels whose boundaries tend to lie on object boundaries within an image. We believe there are many possible avenues for future research in combining the strengths and weaknesses of both approaches. For instance, one weakness of the method of Avidan and Shamir [20] is that if there are discontinuities in the boundaries of objects, removing paths using their cost function can lead to observable errors in the retargeted image. On the other hand our energy function naturally leads to boundary continuity by linking strong edge fragments along a minimum

path across the image. Boundary continuity is found to be the most informative measure of a good segmentation in the work of Ren and Malik [218] and it may be possible to use minimum cost paths in the inverted energy to help reduce retargeting artefacts.

3.11.2 Evaluation methodology

We have demonstrated the use of several different evaluation measures and shown the advantage of using more than one. In many areas of computer vision the research producing new algorithms greatly exceeds work on their evaluation [83] and while the evaluation is thorough there is scope for improvement. For instance the parameters used in evaluating algorithms are chosen heuristically (those that appear to give reasonable segmentations or where possible recommendation from authors) whereas it may be possible to improve performance, including performance of our method, by searching the parameter space more exhaustively. One method for doing this is suggested by Everingham et al. [86] where they use a genetic algorithm to efficiently search the large parameter space of competing segmentation algorithms but other approaches include training the boundary detection algorithms for each dataset separately or at least on a combined training set. This is likely to improve performance on both the CamVid and VOC dataset.

In a subsequent publication, Everingham et al. [83] also use probabilistic analysis of the behaviour of algorithms in the joint fitness/cost space. While our approach of using the standard error goes someway to characterizing the uncertainty in the performance of different algorithms it is clear that this analysis could be extended to show how frequently and algorithm is likely to produce a very good or very poor segmentation result. A strength of our algorithm is that it is constrained in such a way that it provides a reasonable result on most images and this is not currently emphasized in our analysis.

Furthermore, we have found it useful to summarize results across different measures, resolutions and datasets using the Borda score. The Borda score was used as a summary statistic because of its simplicity and ease of calculation when aggregating partial ranked lists. We also feel that it manages to capture the trade off between competing algorithms. However, it is only an approximation technique for the *partial rank aggregation* problem which is NP-hard [79]. The problem has received consid-

erable attention in recent years as a result of producing ranking functions for search engines [79]. There are several other popular distance measures on a set of partial rank lists including Spearman's footrule distance [245] and Kendall's τ distance [133] and it would be worth exploring the merits of different measures in future work. One property of particular interest would be the sensitivity of algorithm ranking to the particular set of measures used during evaluation - ideally we would like to start with a large set of measures whose merits it is difficult to judge and then show that using only a small subset of these measures it is possible to obtain the same rankings as the results on the full fitness/cost space.

3.11.3 Limitations of the algorithm

First, our algorithm divides the image into a series of overlapping strips. This imposes a regularizing constraint but it is not data dependent. This can cause reduced performance on certain images if all of the object data is in a small region of the image. One solution to this is to use a high resolution lattice, so that there are enough superpixels in the correct region of the image, and then merge down, but this will not always have the desired effect. In the next chapter we shall see how to distribute the strips in the image in a more principled manner.

Second, our method depends on a boundary map which has both advantages and disadvantages. The boundary map can be expensive to compute and a reasonably good map is required to achieve good performance [182]. We have made use of work on boundary detection algorithms [43, 174, 74] and have not investigated the properties of the boundary map itself. For instance, there is no need for the mapping β used for the boundary map to be linear nor for β to be isotropic or stationary across image. It may be possible to alter the boundary map dynamically - in a similar manner to the path constraints - in order to reduce the types of artifacts that are specific to our lattice method. It may also be beneficial to alter the cost of particular types of edges corresponding to gestalt cues - eg. straight or parallel lines or constant high curvature - that may effect the performance of our lattice method. For instance, path length considerations mean there is a preference towards boundaries that go up-down or left-right in the image rather than on the diagonal. It may be possible to alter the boundary map, or use different boundary maps for different orientations of strips, to help mitigate

this fact.

Third, both of our graph constructions impose limitations on the method. There is the limitation of solely depending on a boundary map and not explicitly using region information - we shall return to this problem in Chapter 5. We have also not exploited additional properties of the first solution for finding minimum paths. The G_{dag} graph construction has the advantage of lower complexity but also the possibility of using negative edge weights. These might have a use in certain applications where we want to reduce the weight of edges dynamically without re-scaling all edges in the graph.

Finally, we have not explored the merging properties of our method in detail. It is likely that we could boost performance of `NC` and `Uniform` algorithms in a similar merging scheme to that presented in this chapter. Hierarchical mean shift has been explored in the work of Paris and Durand [202] where they also provide techniques to speed up the algorithm at high spatial bandwidths. Understanding the relationship between the different modes of failure of agglomerative and divisive methods is left for future work.

3.12 Conclusions

In this chapter we have put forward a series of arguments to support the claim that a number of desirable properties of pixels should be maintained by superpixels and that this is not possible with existing algorithms. We have developed an algorithm that satisfies some of these additional properties and demonstrated good performance on standard datasets.

We can see from the comparative analysis of different datasets (Section 3.5.4) that there are differences in performance across all algorithms. Moreover, we have highlighted several situations where the design of our algorithm makes it difficult to produce good results.

In the next chapter we investigate methods for incorporating information about the likely distribution of objects within a scene to improve the performance of superpixel algorithms.

Chapter 4

Scene Shape Priors

In the last chapter we demonstrated a regular lattice of superpixels can support good segmentation performance. However, we also highlighted some of the problems associated with it and noted a difference in performance across datasets. In this chapter we exploit prior knowledge of the distribution of objects within a scene to adapt the distribution of superpixels to improve performance.

4.1 Introduction

All the methods of over-segmentation that we saw in Chapter 2, and the greedy regular lattice that we introduced in the last chapter, are bottom-up processes. An alternative approach that is commonly encountered in vision pipelines is a *top-down* approach. This approach uses an object representation learned from exemplars to aid the process of segmentation [126, 33, 143, 282, 31, 157] but it is not straightforward to adapt these methods for over-segmentation.

Despite this, there have been several interesting approaches to help improve upon an initial over-segmentation. These include combining information from multiple segmentations [121, 172, 139] or iteratively revising the set of regions based on progressive updates of the interpretation of the scene [120]. However neither of these approaches actually alters the process of over-segmentation itself — they simply attempt to mitigate some of the difficulties associated with over-segmentation using additional steps in the processing pipeline.

On the other hand, recent progress in related areas of vision research such as object detection and recognition have successfully exploited prior information to help improve results. Examples include: object priors [260, 119], scene category priors [116, 162]

Figure 4.1: Missing cars example. a) Original CamVid image with inset showing small cars. b) GRL algorithm produces a poor segmentation by missing cars at a distance.

and region priors [115, 240].

In contrast there has been very little work on incorporating priors in the process of over-segmentation. For instance while it is noted in [88] that it is possible to have the segmentation method prefer regions of a certain size or shape this is not exploited. In [185] uniform size superpixels are encouraged using postprocessing regardless of where they appear in the image. Similarly, the uniform distribution of strips in the algorithm we presented in the last chapter implicitly imposes a quasi-uniform segmentation on the image by two mechanisms: First, the strips are distributed evenly throughout the image leading to a roughly even distribution of paths. Second, the minimum cost path measure favours shorter paths across the image which are consequently relatively straight.

This quasi-uniform distribution of superpixels is sensible if there is no *a priori* knowledge of the distribution of objects within the image. However, this is generally not the case and for many classes of image we hypothesize that non-uniform sampling would be beneficial. For example, when a 2D image is a projection of a 3D scene, perspective effects commonly result in an uneven distribution of the sizes of classes. Figure 4.1 gives one example of a poor segmentation using the method we presented in the last chapter. The small cars are missed (the region is labeled building using the ground truth) because they are very small in the original image. In general we suggest that we should try and learn something about the expected difference in the sizes of classes and their distribution across the image to help improve the segmentation.

The work in this chapter sets out to address this issue. In Section 4.2 we show that the density of object boundaries varies according to the distribution of objects in the scene. In Section 4.3 we alter the algorithm presented in the previous chapter so that it partitions an estimate of this density roughly equally between superpixels whilst still attempting to capture local object boundaries. Section 4.3 then introduces a probabilistic model that describes the spatial density of object boundaries within an image. We demonstrate that our algorithm successfully learns this foveated [1] spatial distribution and can exploit this knowledge to improve segmentation performance. In the following section we investigate the distribution of ground truth regions between different datasets and use this to motivate the work presented in the rest of the chapter.

4.2 Distribution of Ground Truth Regions

We saw in Chapter 2 that the average number of regions per image varies between datasets. Here we investigate the distribution of regions in ground truth data to motivate the work in this chapter.

Figure 4.2 and Figure 4.3 show the distribution and size of regions in the ground truth data for the different datasets. Plots on the left hand side of the Figures show the centroid of regions in the ground truth (normalized image co-ordinates) with a circle indicating the size of the region (quantized into a hundred bins). In Figure 4.2a we can see that BSD has a fairly even distribution of regions across the image with larger regions appearing in the center of the image. This distribution of regions may result from the original instructions to the subjects who segmented the data:

> *Divide each image into pieces, where each piece represents a distinguished thing in the image. It is important that all of the pieces have approximately equal importance.*

This instruction results in ground truth regions that include objects parts and textural details across the image.

In contrast the VOC dataset (Figure 4.2c) tends to have a dominant large central region with far fewer small regions on the periphery. Additionally, the increased number of large image regions can be seen from the larger tail on the right hand side of

[1] Here we use the term "foveated" to indicate that the image resolution varies across the image according to one or more "fixation points" – ie. the distribution can be multi-modal.

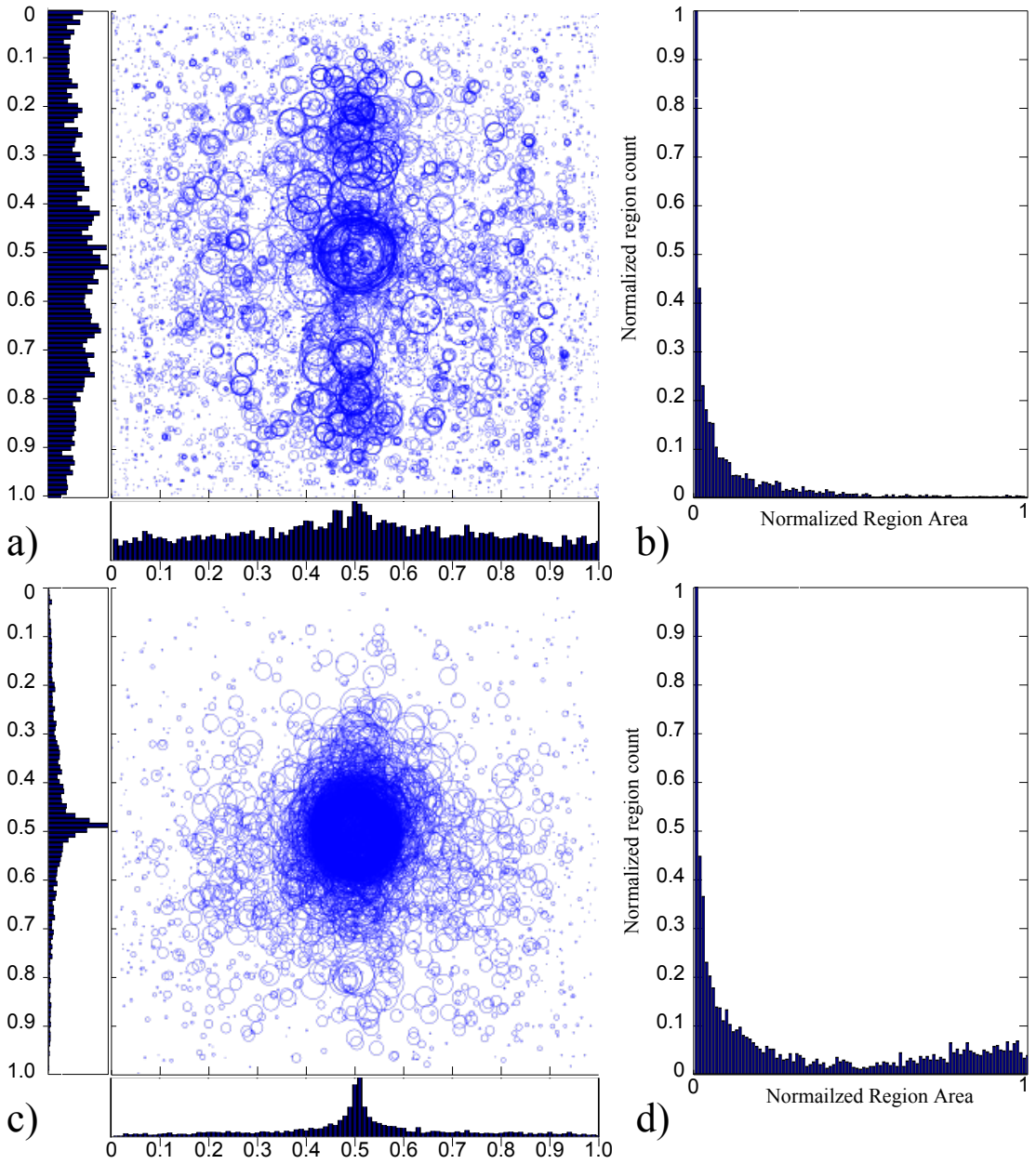


Figure 4.2: Plots to show the size and distribution of dataset ground truth regions. Plots in the first column mark the centroid of regions in the ground truth with a circle based on the size of the region (normalized image co-ordinates). Normalized histograms show marginal distribution of centroids. Second column shows distribution of region sizes normalized by area of the image. a-b) BSD [174] c-d) VOC [87]. Note the different spatial distributions of regions between datasets.

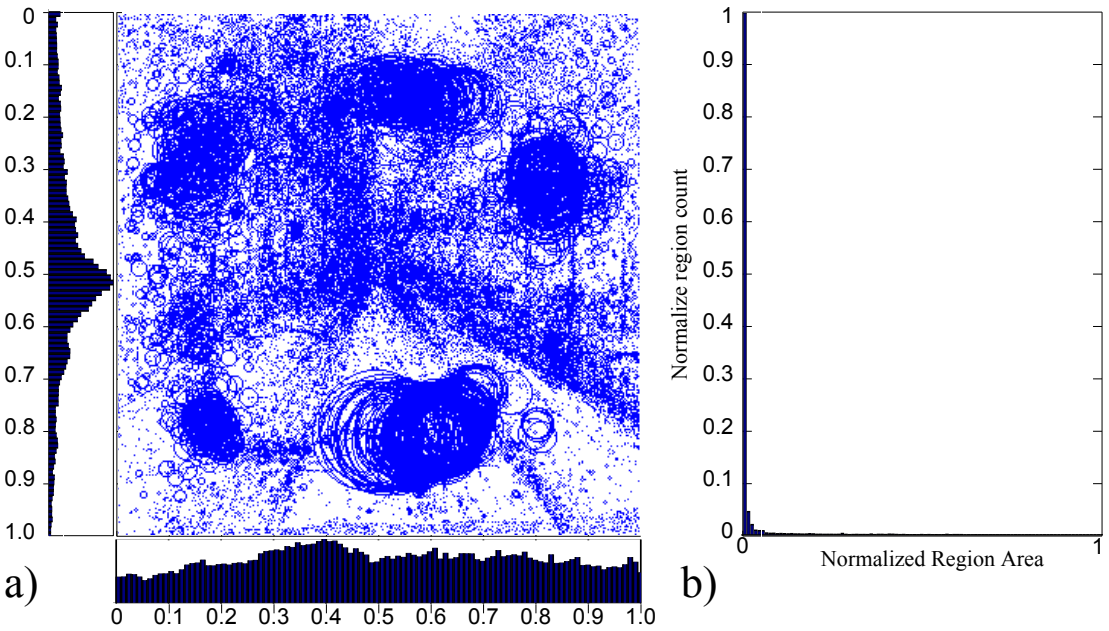


Figure 4.3: Plots to show the size and distribution of CamVid [39] dataset ground truth regions. a) Centroid of regions in the ground truth marked with a circle based on the size of the region (normalized image co-ordinates). Normalized histograms show marginal distribution of centroids. Note the distribution on the Y-axis is multimodal and exhibits more structure than dataset in Figure 4.2. Long thin clusters at 45 degrees demonstrate the left hand driving bias with road markings and pavement features. b) The distribution of region sizes normalized by area of the image.

Figure 4.4: Prototypical views from CamVid dataset [39]. a) Hand selected“Turning manoeuver” images b) Hand selected “Frontal driving” images.

the normalized histogram of region size (Figure 4.2d). Although the VOC images are collected from photo-sharing website which results in an "unbiased" dataset the greater propensity for the object(s) to be large and in the center of the image is probably due to object(s) of interest being the subject of the photograph.

On the other hand the distribution for the CamVid dataset (Figure 4.3a) looks markedly different. This dataset exhibits a distribution of regions with more structure. Large regions appearing in the center top (sky) and center bottom (road), although unlike the large regions in the VOC dataset they are not objects of interest centered in the image. Additionally, we can see from the vertical histogram (RHS Figure 4.3a) that there is a large distribution of small regions in a horizontal band across the center of the image. This distribution of ground truth regions occurs because it is common for vehicles and pedestrians at a distance, near the vanishing point, to appear in small clusters in the center of the image.

This observation leads to two questions: 1) "How should we predict the likely distribution of objects within an image?" and 2) "How should the information on the distribution of objects be used to influence a superpixel algorithm?" Our approach to these two questions in set forth in the following sub-section. The results presented in Section 2.5.3 and Section 3.5.4 and the evidence in Figures 4.3 suggest that there is more to be learned about the distribution of objects within the CamVid data and we make this dataset the focus of the chapter.

4.2.1 Scene Shapes

The method presented in the last chapter exploited an estimate of object boundaries within an image — the boundary map — which was used to influence the shape of paths though strips of the image. We make two observations about the distribution of object boundaries in images.

First, the distribution of boundaries is based on objects of similar size distributed on a horizontal ground plane - small clusters of objects at a greater distance from the camera will increase the density of object boundaries over a unit area of the image. This is illustrated in Figure 4.5a. In regions of high boundary density we require a greater number of superpixels to potentially capture the increased number of objects within a fixed region of the image.

Second, the boundary density varies spatially between images, not only because the layout of objects varies between images but because of the scene geometry. For example, we noted that small objects in the ground truth of the CamVid dataset can appear in a narrow region in the center of the image. However, it is also easy to find examples where this is not the case. Images in Figure 4.4 have been selected to illustrate two prototypical views from within the CamVid dataset. The first set of images (4.4a) are referred to as "turning manoeuvres" where the vehicle approaches a frontal planar surface and the viewpoint can be characterized by two point perspective (see Figure 4.5b). The second set of images (4.4b) are referred to as "frontal driving" where the vehicle is in motion on the LHS of the road and the viewpoint can be characterized by one point perspective (see Figure 4.5c).

We refer to the varying distribution of object boundaries within a scene as the *shape* of the scene. Furthermore, we shall assume that knowledge of this 'shape' can be captured in the form of a *boundary distribution image* (BDI): each pixel takes a value between 0 and 1 representing the prior probability of observing a real-world boundary at this position (see Figure 4.6). The model for the BDI is parameterized so it contains global information about the the whole dataset but also such that it can adapt to the particular distribution of boundaries in each new image - it therefore provides *a priori* information on the likely distribution of boundaries in an image. We therefore refer to the information contained in the BDI as the *boundary distribution prior*. The goal of Section 4.3 is to show how to exploit the BDI to improve segmentation in the superpixel lattice algorithm. To do this we first assume that we know the model for the BDI and can estimate the distribution of boundaries in images. We then present a model and learn its parameters from training data in Section 4.4.

4.3 Adaptive Regular Lattices

To exploit the boundary distribution prior in the superpixel lattice algorithm we must solve two problems: 1) We must adapt the position and shape of the strips so that each has approximately the same prior probability of containing a boundary. 2) We must adapt the minimum cost path algorithm so that the best path will tend to follow the shape of the strips. We present two different approaches. The first approach (Sections 4.3.1 and 4.3.2) solves each problem separately. The second approach (Section 4.3.3)

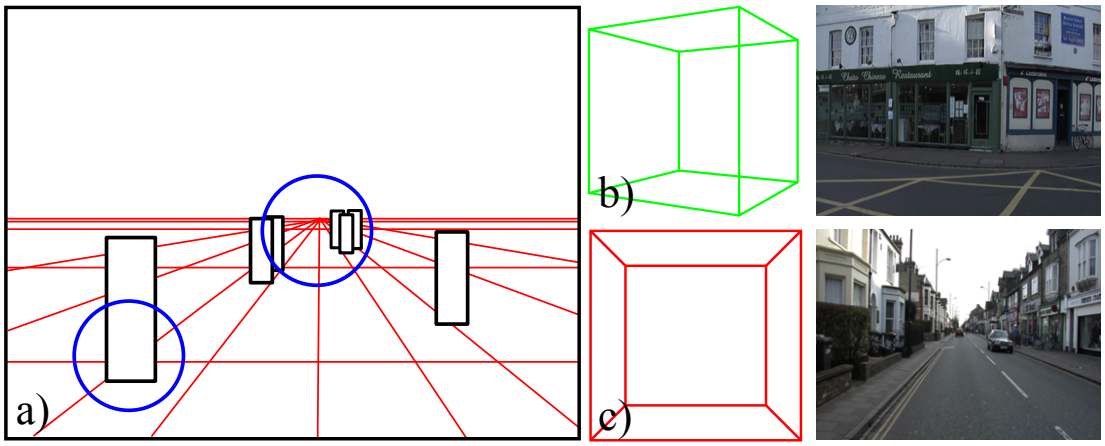


Figure 4.5: Influence of perspectives. a) Similar sized object on a horizontal ground plane. There are twice as many boundaries in the central blue circle - containing five objects - than the left circle - one object. b)Two point perspective of "turning manoeuvres" - two vanishing points c) One point perspective of "frontal driving" - one vanishing point.

combines both in a single procedure.

4.3.1 Non-uniform Strips

We aim to calculate a set of strips across the image so that there is an equal chance of finding a boundary within each. We first discuss the assignment of vertical strips. For each row of the image we calculate the cumulative probability of observing a boundary as we move from left to right. This can be computed by integrating each row of the boundary distribution image and dividing by the total probability mass for that row. The result is a normalized cumulative distribution image (Figure 4.6b).

We now allocate strips so that they partition the cumulative distribution equally. In practice, this means that the edge of the strips follow the iso-contours of the normalized cumulative distribution image. For horizontal strips, we integrate the boundary distribution image in the vertical direction and normalize. We allocate strips that follow the iso-contours of this vertical normalized cumulative distribution. The result is a set of strips that are non-uniform - they may meander through the image and vary significantly in width depending on the position.

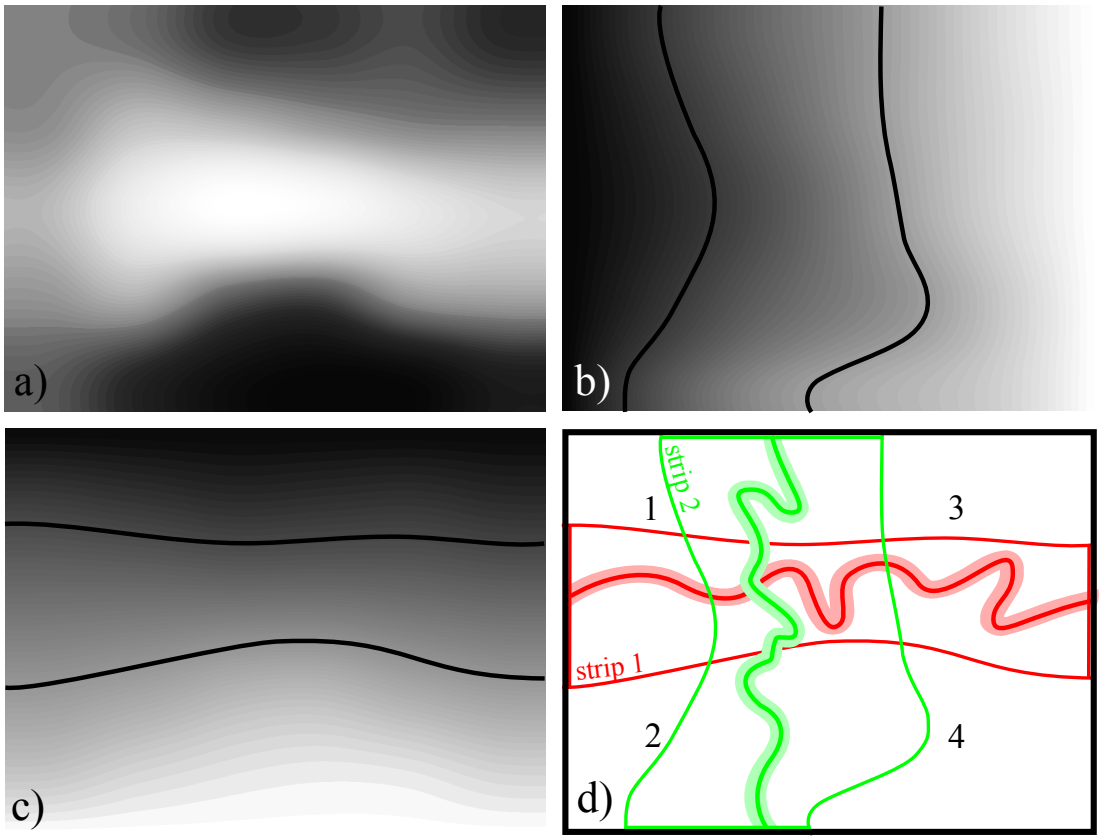


Figure 4.6: Construction of non-uniform strip a) Smoothed boundary distribution prior representing probability of observing boundary at each position. b) Normalized cumulative distribution in horizontal direction. c) Normalized cumulative distribution in vertical direction. d) Boundaries of strips now follow iso-contours in these integral boundary maps. Compare with Figure 3.4a.

4.3.2 Non-uniform Minimum Cost Path Algorithm

We now seek to find an optimal path for each strip that remains within the strip boundaries and bipartitions the image. The `GRL` algorithm used Dijkstra's algorithm to find the minimum cost path across the strip and the cost was determined by the boundary cost map. Hence, the algorithm tends to follow boundaries as these are inexpensive, but also aims to minimize path length. Unfortunately, this presents a problem with non-uniform strips. The path tends to take the straightest route across the image (Figure 4.7a) rather than follow the shape of the strip (Figure 4.7b). This effectively means that the superpixel size will not vary according to the prior.

To solve this problem we take the following approach: we warp the strip so that the average path along the warped version is now straight. We find the minimum cost

path along the warped strip. Finally, we unwarp the path back to the original space. In this way we effectively define a distance metric along the strip so that shorter distances follow the shape of the boundaries (see Figure 4.7 b-d-f).

For a horizontal strip, the warping is achieved by applying a separate one dimensional affine warp to each column. The warps are chosen so that the strip boundaries in that column are always mapped to the same position. The inverse warp consists of applying the inverse one dimensional affine transform to each column.

By repeatedly finding non-uniform strips and finding optimal paths through these strips, it is possible to segment the image in such a way that it is influenced by the boundary distribution image. In particular, the superpixels will be smaller and more densely packed in regions where we expect to find more image boundaries.

4.3.3 Scene shape warping

In a previous publication of this work [183] we use the method of warping each strip separately, just described. However it is also possible to apply the warp to the *whole-image*. In the results presented in this chapter we make use of this alternative approach so that we can use the publicly available implementation of the `GRL` algorithm (written in C++). Moreover, as the warp is applied once to the whole image, rather than each strip, for the experiments carried out on multiple resolutions this method has the advantage of being faster than warping each strip.

This *whole image warp* is based on cumulative image co-ordinates. Cumulative image co-ordinates are used to describe the position of a new co-ordinate based on its cumulative contribution to the BDI. This co-ordinate system is illustrated in Figure 4.8. For example, we begin with a pixel value 1 in the top left hand corner of the toy image, and the pixel location in the original image co-ordinate is (1/3,1/3). The value 1 of the pixel corresponds to 1/4 of the cumulative BDI in the first row and 1/5 in the first column. Therefore, the cumulative image co-ordinate for the pixel is (1/5,1/4). If we do this for every pixel we are left with a set of values (the original pixels) based on irregular sampled data and the warp consists of interpolating a normal pixel grid based on these cumulative co-ordinates. To perform this interpolation we use a nearest neighbour approximation based on Delaunay triangulation using the function `griddata.m` in Matlab which in turn uses the `QHull` algorithm [21].

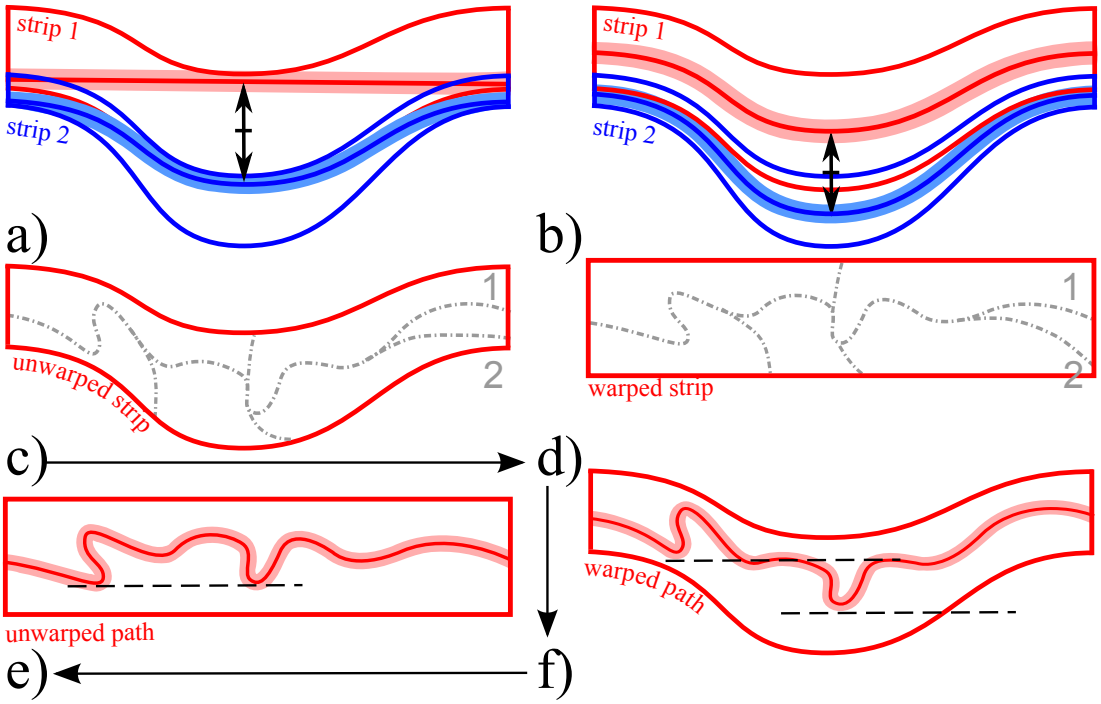


Figure 4.7: Warping minimum cost paths. a) Minimum cost path in unwarped strip. In the absence of boundary information paths go straight across strip. b) Minimum cost path in warped strip. Paths will follow the contour of the strip. Notice the difference in position and size of a superpixel, marked by an arrow, generated in warped and unwarped versions. c) Unwarped strip of boundary cost map. Note here that the shortest path takes branch 2 of the fork. d) Warped strip. In this strip the shortest path takes branch 1 of the fork. e) Minimum cost path in warped strip. f) Warped minimum path in unwarped strip. To apply this to a new image we follow the stages c-d-f-e, giving us a path in the original image that is found in the warped strip.

Similarly to warping the strips this method therefore consists of two warps: The first warp or 'forward warp' takes the data from the cumulative image co-ordinates and produces a warp based on interpolating new data on a regular sampled grid. The forward warp is illustrated in Figure 4.9a. The `GRL` algorithm is run on the boundary cost map in this warped image and returns a regular lattice of superpixels. The second 'backwards warp' then takes this regular sampled data and calculates the unwarped superpixels based on the original cumulative image co-ordinates. The backwards warp is illustrated in 4.9b.

An example of this warping process applied to the BDI itself can be seen in Figure 4.10. We being with an example BDI (Figure 4.10a) and calculate cumulative image

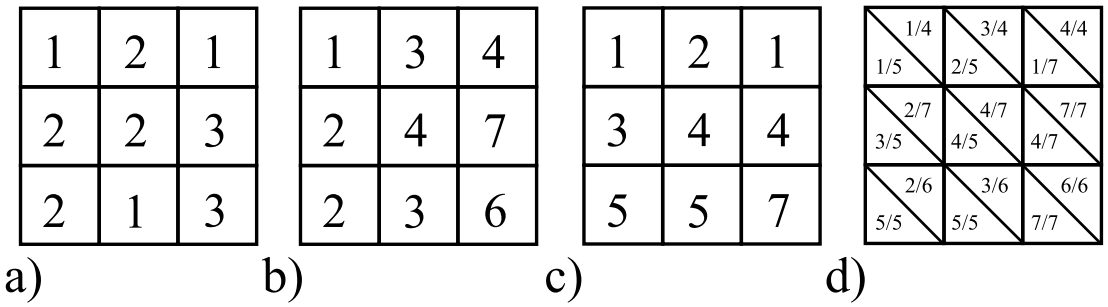


Figure 4.8: Cumulative Image Co-ordinates. a) Toy boundary density image (BDI). b) Row cumulative BDI. c) Column cumulative BDI. d) Cumulative Image Co-ordinates. Top left pixel with original co-ordinate of (1/3,1/3) in normalized image co-ordinates now has co-ordinate (1/5,1/4) in cumulative image co-ordinates.

co-ordinates for each pixel. These new grid points are shown in Figure 4.10c. We then interpolate a new image (Figure 4.10d) based on a regular grid shown in Figure 4.10b. In Figure 4.10d we have shown the warped BDI pixels to make comparison with the original BDI possible, but in practice we are warping the boundary cost map used by the `GRL` algorithm.

A further example of a warp applied to the original image pixels can be seen in Figure 4.11. This shows intuitively what the warp is trying to achieve - regions of the image that have a high density in the BDI are expanded and regions in low density regions are contracted. In the example in Figure 4.11 this has expanded the pedestrian and cars at a distance and has reduced the size of the road and sky classes. Further examples can be seen in Section 4.5. The method presented in this section allows the BDI to be applied to general superpixel algorithms rather than just the strip based `GRL`.

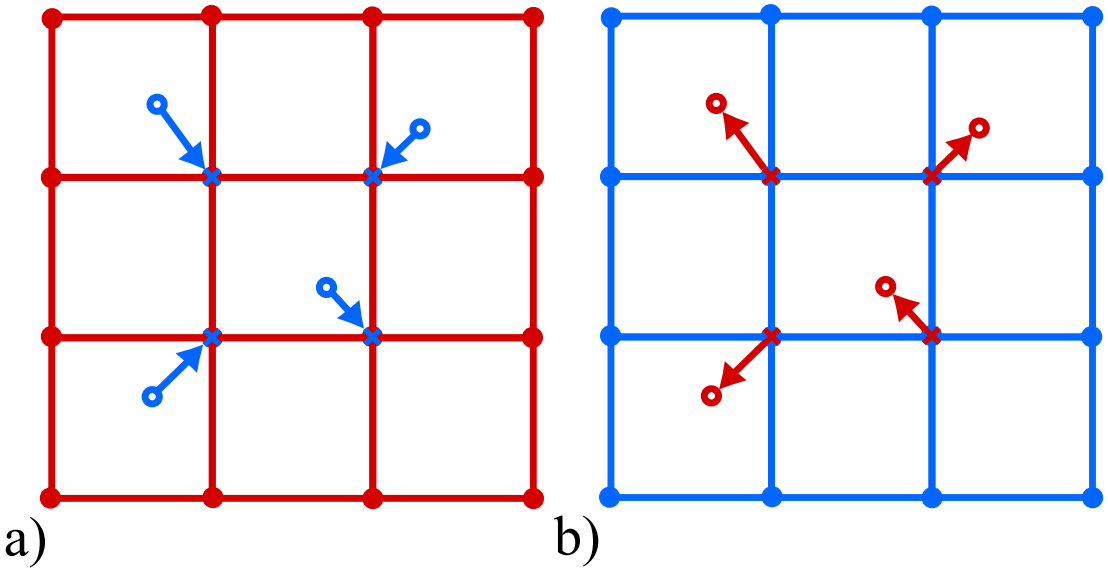


Figure 4.9: Interpolating warped data points. a) Forward warp. We begin with a set of data based on an irregular sampled grid (blue circles). These are pixel values based on cumulative image co-ordinates. We then interpolate new values on a regular sampling (red grid). This has the effect of warping our original image based on the BDI. b) Backwards warp. We begin with regular sampled data (blue circles) and interpolate the irregular sampled data based on the cumulative image co-ordinates (red circles).

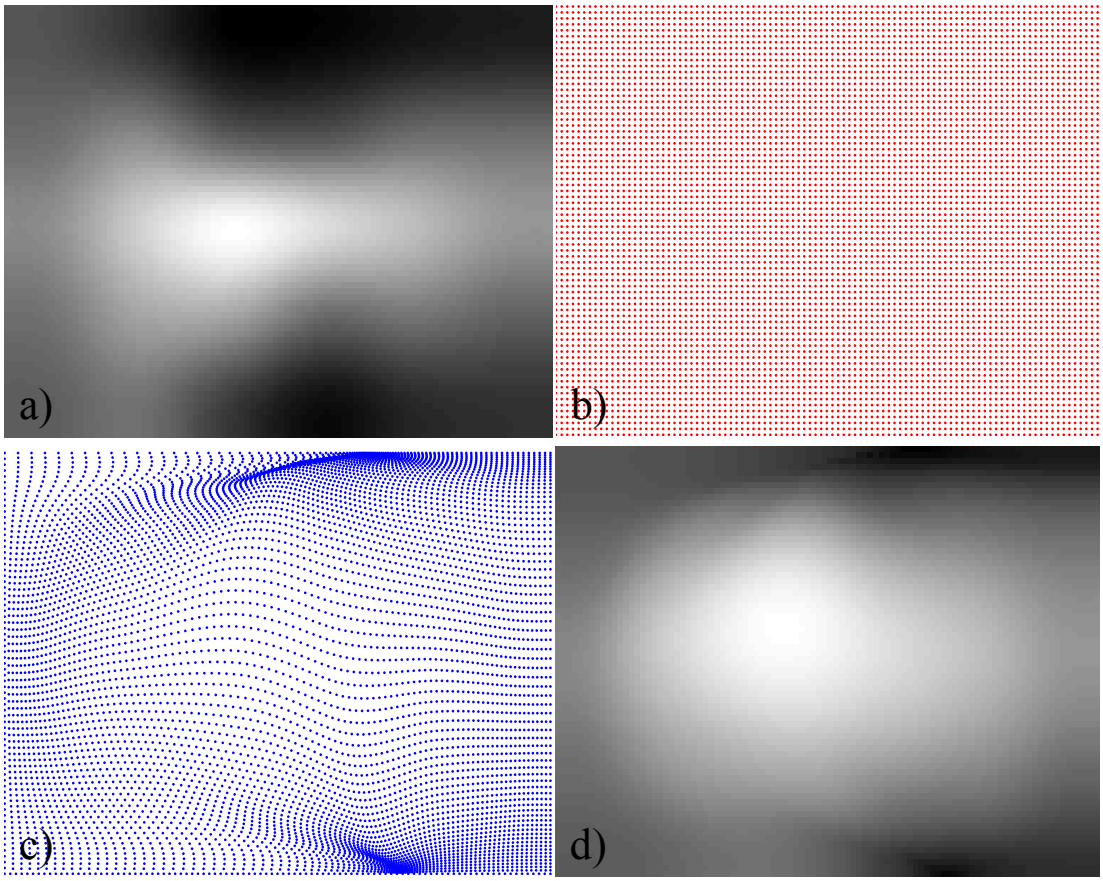


Figure 4.10: Warping Boundary Distribution Image. a) Original BDI. b) Regular sampling grid. c) Grid based on cumulative image co-ordinates. d) Example of warping BDI [a] based on sampling grid in [c]. Notice that the high density region in the middle of the original BDI [a] is spread both upwards and downwards and to the right in the new image. In practice we warp the Boundary Cost Map, not the BDI.

Figure 4.11: Comparison of warped images. a) Example boundary distribution image (BDI). b) Original ground truth data. c) BEL boundary cost map. d) Original pixel image with regions of interest marked with coloured boxes e) Warped BEL boundary cost map. f) Forward warped image. Note the change in area of selected classes between the two images. LHS pedestrian (red box) is now larger whilst empty road is compressed (green box).

4.4 Boundary distribution prior

Until now we have assumed that we know the boundary distribution image (BDI). In this section we describe an algorithm to take an observed set of images and learn a model which describes the BDI with a small number of parameters. When we see a new image we fit these parameters to estimated the BDI for this image. The image will depend on both prior information about the spatial statistics of boundaries (learnt from training data) and the observed edge data from a new image. In this section we present the model before describing learning (Section 4.4.1) and inference (Section 4.4.2) algorithms.

We wish to describe a probability distribution over observed boundaries $\mathbf{x} = [x_1 \ldots x_P]^T$ at the p pixels of an image. Each element x_p is binary and is 1 when a boundary is present and 0 when it is absent. We assume that x_p is drawn from a single observation of a Bernoulli distribution with parameter y_p. Our goal then is to model the joint probability distribution of the vector of Bernoulli parameters $\mathbf{y} = [y_1 \ldots y_P]^T$. The vector $\mathbf{y}$ represents the boundary distribution image or BDI.

We assume that the distribution over $\mathbf{y}$ is determined by an underlying mixture of K clusters with each cluster having a subspace representation. We term this a "clustered latent trait" or CLT model [24]. More precisely, we assume that associated with image $\mathbf{y}_i$ there is (i) a discrete hidden variable c indicating which of K clusters generated the data and (ii) a continuous hidden variable $\mathbf{h}$ that represents the position within the subspace associated with that cluster. The variable $\mathbf{h}$ weights J basis functions $\mathbf{f}_{1k} \ldots \mathbf{f}_{Jk}$ that form the columns of a matrix $\mathbf{F}_k = [\mathbf{f}_{1k} \ldots \mathbf{f}_{Jk}]$ associated with the k'th cluster.

We define the activation $\mathbf{a} = [a_1 \ldots a_P]^T$ for the i'th image to be a vector representing the propensity of each pixel to contain a boundary and calculate it as $\mathbf{a} = \boldsymbol{\mu}_c + \mathbf{F}_c\mathbf{h}$ where $\boldsymbol{\mu}_c$ is a mean vector that describes the average activation for cluster c. The activation $\mathbf{a}$ contains numbers defined on the whole real axis, and we convert these to a probability $\mathbf{y}$ by passing each element corresponding to pixel p through the logistic sigmoid function:

$$y_p = \sigma(a_p) = \frac{1}{1 + \exp(-a_p)} \tag{4.1}$$

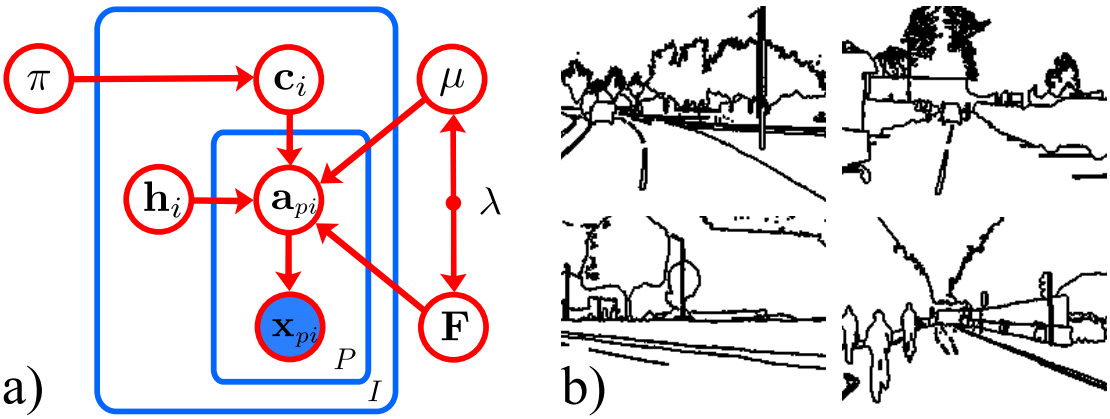


Figure 4.12: Graphical model. a) Clustered Latent Trait Model in which the plate denotes a set of I images each with P pixels. Here $\mu = \{\mu_k\}$, $F = \{F_k\}$. b) Example training data adapted from the CamVid dataset [39].

Finally, we assume that the probability of observing a boundary x_p at pixel p is given by y_p. We can summarize this model concisely as:

$$Pr(c = k) = \pi_k \tag{4.2}$$

$$Pr(\mathbf{h}) = Norm_{\mathbf{x}}[\mathbf{0}, \mathbf{I}] \tag{4.3}$$

$$Pr(\mathbf{a}|\mathbf{h}, \mathbf{c}) = \delta_{\mathbf{a}}(\boldsymbol{\mu}_c + \mathbf{F}_c\mathbf{h}) \tag{4.4}$$

$$Pr(\mathbf{x}|\mathbf{a}) = \prod_{p=1}^{P} \text{Bin}_{\mathbf{x}_p}[\sigma(a_p)] \tag{4.5}$$

where $Norm_\alpha[\beta, \mathbf{\Gamma}]$ represents a Gaussian in variable α with mean β and covariance Γ. The function $\delta_\alpha(\beta)$ denotes a probability distribution over α where all of the mass is at β and hence describes a deterministic relationship. The function $\text{Bin}_\alpha[\beta]$ denotes the binomial likelihood of observing value α given binomial parameter β. The term π_k represents the prior probability of choosing the k'th cluster and in Equation 4.3 we have also defined a prior over $\mathbf{h}$. This graphical model is illustrated in Figure 4.12.

4.4.1 Learning

Our goal is to learn the parameters $\theta = \{\pi_{1...k}, \mu_{1...k}, \mathbf{F}_{1...k}\}$ of the CLT model based on I binary training images $\mathbf{x}_{1...I}$ where the value at each pixel represents the presence of a boundary. In particular we will maximize the joint log likelihood of all of the variables

$$L = \sum_{i=1}^{I} [\log Pr(\mathbf{x}_i|\mathbf{h}_i, c_i, \mathbf{F}_{1...K}, \mu_{1...K}) + \log Pr(\mathbf{h}_i) + \log Pr(c_i)] + \sum_{k=1}^{K} [\log Pr(\mathbf{F}_k) + \log Pr(\mu_k)] \tag{4.6}$$

where we have defined Gaussian priors over the matrices of basis vectors $\mathbf{F}_{1...K}$ and the means $\mu_{1...K}$ so that

$$Pr(\mathbf{F}_k) = \prod_{j=1}^{J} Norm_{\mathbf{f}_{jk}}[\mathbf{0}, \lambda\mathbf{L}^{-1}] \tag{4.7}$$

$$Pr(\mu_k) = Norm_{\mu_k}[\mathbf{0}, \lambda\mathbf{L}^{-1}] \tag{4.8}$$

where $\mathbf{f}_{jk}$ is the j'th column of matrix $\mathbf{F}_k$, $\mathbf{L}$ is the discrete approximation to the Laplacian operator and λ is a constant that was set by hand and controls the influence of the prior. These priors encourage spatial smoothness.

To describe the learning algorithm, first assume that the discrete variable $c_{1...I}$ representing the cluster assignments for each image are known. For each cluster, we use a strategy where we alternately maximize L with respect to the hidden variables $\mathbf{h}_i$ and the parameters θ. After several iterations, we reassign cluster assignments by finding the cluster $c_i \in \{1 \ldots K\}$ under which the data $\mathbf{x}_i$ is most likely. This conditional likelihood is calculated using the optimal value of the hidden variable $\mathbf{h}_i$ for each cluster. We also re-estimate the prior probability $\pi_{1...K}$ of each cluster. Having reassigned the points, we then relearn each cluster separately and so on. Algorithm 3

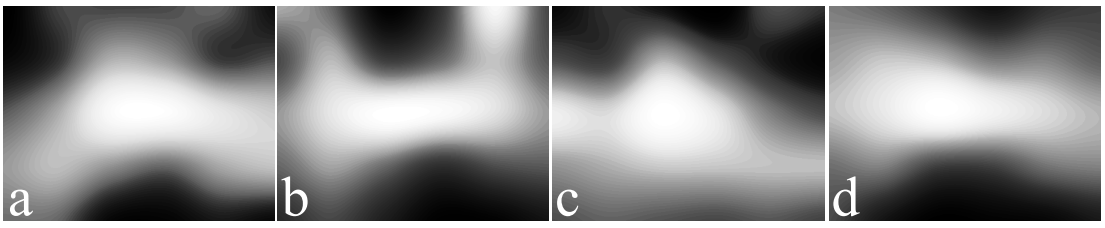


Figure 4.13: Learned cluster means. Each mean has a strong peak in the center of the image (a common boundary distribution for road scenes) but there are subtle changes in the distribution around this point.

Algorithm 3 Learn Clustered Latent Trait Model

for $t_1 = 1$ to T_1 **do**
  // Reassign data to clusters
  for all i **do**
    $c_i, \mathbf{h}_i \leftarrow \arg\max_{c_i,\mathbf{h}_i} \log Pr(\mathbf{x}_i, \mathbf{h}_i, c_i, \theta_i)$
  end for
  // Update parameters for each cluster
  for $t_2 = 1$ to T_2 **do**
    for all k **do**
      $\theta_k \leftarrow \arg\max_{\theta_k} \sum_{s\in\{s:c_s=k\}} \log Pr(\mathbf{x}_s, \theta_k, c_s, \mathbf{h}_s)$
    end for
    for all i **do**
      $\mathbf{h}_i \leftarrow \arg\max_{\mathbf{h}_i} \log Pr(\mathbf{x}_i, \mathbf{h}_i, \theta, c_i,)$
    end for
  end for
end for
return $\theta_{1\ldots K}$

describes this process more formally. The maximization over the discrete parameters c_i was done exhaustively. The maximization over the continuous parameters θ and $\mathbf{h}_{1\ldots I}$ was performed using a quasi-Newton method making use of the following derivatives:

$$\frac{\partial L}{\partial \mathbf{h}_i} = \mathbf{F}^T(\mathbf{x}_i - \mathbf{y}_i) - \mathbf{h} \tag{4.9}$$

$$\frac{\partial L}{\partial \mathbf{F}_k} = \sum_{s\in\{s:c_s=k\}} (\mathbf{x}_i - \mathbf{y}_i)\mathbf{h}_s + \lambda \mathbf{L}\mathbf{F}_k \tag{4.10}$$

$$\frac{\partial L}{\partial \mu_k} = \sum_{s\in\{s:c_s=k\}} (\mathbf{x}_i - \mathbf{y}_i) + \lambda\mu_k \tag{4.11}$$

The update for $\pi_1 \ldots \pi_k$ is calculated in closed form:

$$\pi_k = \frac{1}{I}\sum_{i=1}^{I} \delta(c_i = k) \tag{4.12}$$

For the experiments presented in this chapter we set $J = 3$ and $K = \{1, 4\}$ by hand. Examples of the boundary distribution image $\mathbf{y}$ associated with each cluster mean are illustrated in Figure 4.13.

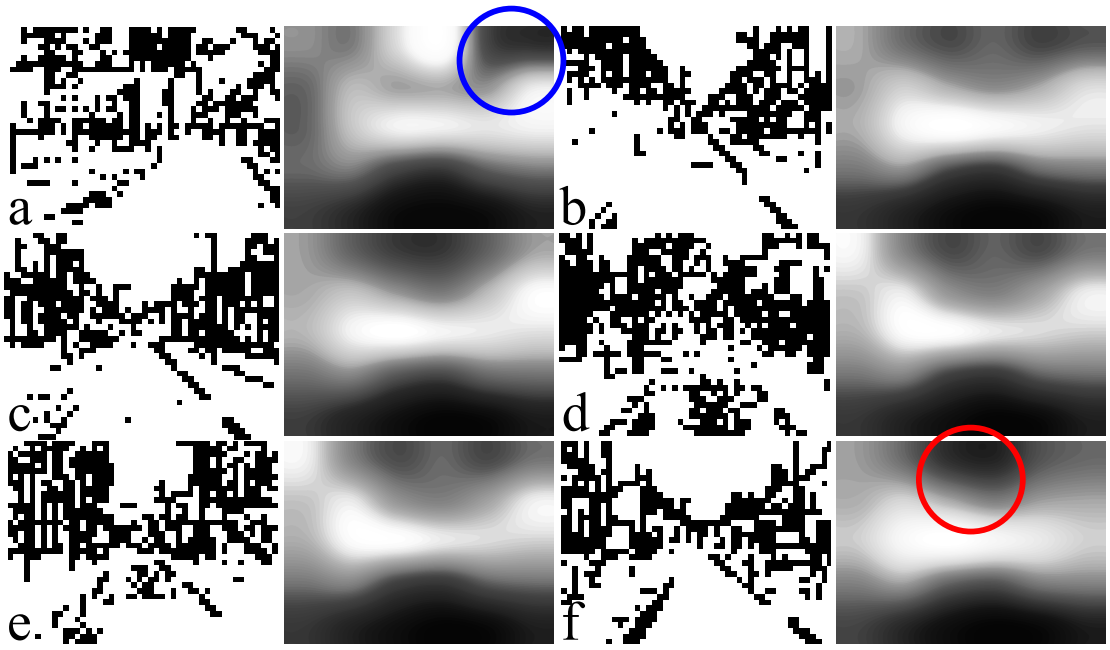


Figure 4.14: Sampling from one cluster during inference. Image pairs consist of output of a unary classifier, $\hat{\mathbf{x}}_\mathbf{t}$, and maximum likelihood estimate of $\mathbf{y}_\mathbf{t}$. Notice the shift from right to left from a) (sky region in blue circle) to f) (sky region in red circle) as the direction of the road changes suggesting the sub-space model is a useful representation.

4.4.2 Inference

In this section we describe how to predict the boundary distribution image that was most likely to have been responsible for a new observed image. We calculate a binary edge map for the new image and use this as a proxy for the unseen boundary map $\mathbf{x}_t$. We then find the the cluster c_t and hidden variable $\mathbf{h_t}$ that were most likely to have created it:

$$c_t, \mathbf{h}_t \leftarrow \arg\max_{c_t, \mathbf{h}_t} \log Pr(\mathbf{x}_t, \mathbf{h}_t, c_t, \theta) \tag{4.13}$$

We use the generative model to calculate the activation $\mathbf{a}_t = \mu_{\mathbf{c_t}} + \mathbf{F}_{c_t}\mathbf{h}_t$ associated with these variables. Finally, the elements of the binomial probability vector $\mathbf{y}_t$ are calculated using Equation 4.1. This process is illustrated in Figure 4.14.

4.5 Quantitative Evaluation

To learn the model presented in Section 4.4 we construct training data from the set of 406 binary ground truth boundary images. In light of the results from the previous chapter, we use the output of the *boosted edge learning* algorithm (BEL) [74]. As the

Metric	F_{sd}						Cover Score					
Lattice Resolution / Algorithm/ Parameters	42	100	210	342	600	900	42	100	210	342	600	900
	6×7	10×10	14×15	18×19	24×25	30×30	6×7	10×10	14×15	18×19	24×25	30×30
ARL-1(BEL)	0.289	0.356	0.437	0.482	0.545	0.588	0.687	0.749	0.799	0.824	0.848	0.862
ARL-4 (BEL)	**0.290**	0.356	**0.438**	**0.484**	**0.546**	0.588	**0.689**	**0.750**	**0.800**	0.824	**0.849**	**0.863**

Table 4.1: Selecting clusters. Performance of setting $K = 1$ and $K = 4$ using the ARL algorithm. ARL-4 shows a small improvement in performance.

spatial prior influences only the distribution of strips, not the final minimum paths, it is sufficient to use very low resolution images and we down-sample from 720×960 to 36×48 using an **OR** operation (inclusive). For inference with a new image we threshold the output of the `BEL` algorithm at 0.5, down-sample and vectorize this binary image to form $\hat{\mathbf{x}}_t$. In the following evaluation we ignore regions smaller than 10 pixels to mitigate some of the variability in the ground truth data. Also when calculating F_{sd} we follow Brostow et al. [40] and base our evaluation on 11 of the most common classes shared between training (06R0,16E5) and test (05VD) sets.

Similarly to the last chapter it is possible to relax the restriction of the fixed topology of the lattice by subsequently merging superpixels. We begin with our *adaptive regular lattice* (`ARL`) algorithm for a fixed number of superpixels by starting with a higher resolution grid and then merging using the Mean Shift algorithm (`ARL-MS`). However we would like to prevent small regions in the center of the image being merged. To achieve this we threshold the BDI at 0.9 and use this as a mask to prevent regions of the image that are assumed to have high probability of a boundary being merged.

For the evaluation we use the same methodology presented in the last chapter. We use six performance measures, interpolate values for six superpixel resolutions where required, and summarize the results using the Borda score. Details can be reviewed in Section 3.5.2 and full results can be found in Appendix B.

4.5.1 Selecting clusters

To show that the cluster model is useful we compare the `ARL` algorithm using $K = 1$ and $K = 4$. Results for two performance measures can be seen in Table 4.1. We can see a modest advantage of using the clustered model over a single factor matrix and we use $K = 4$ in the experiments in the following sections.

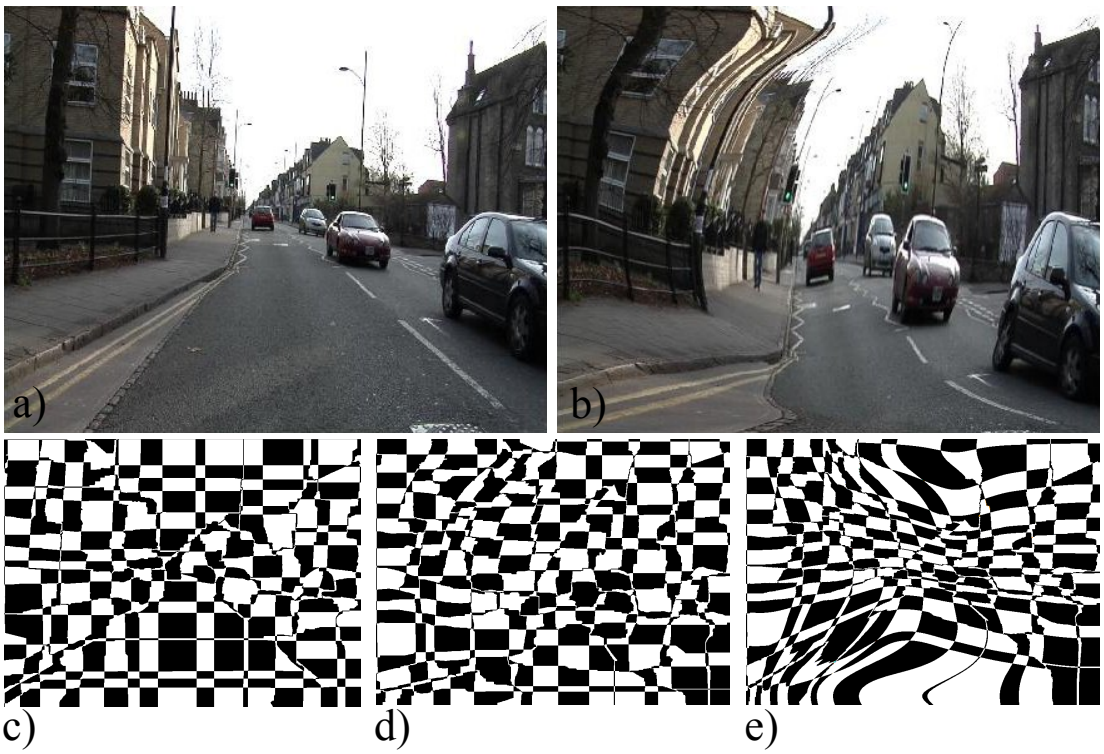
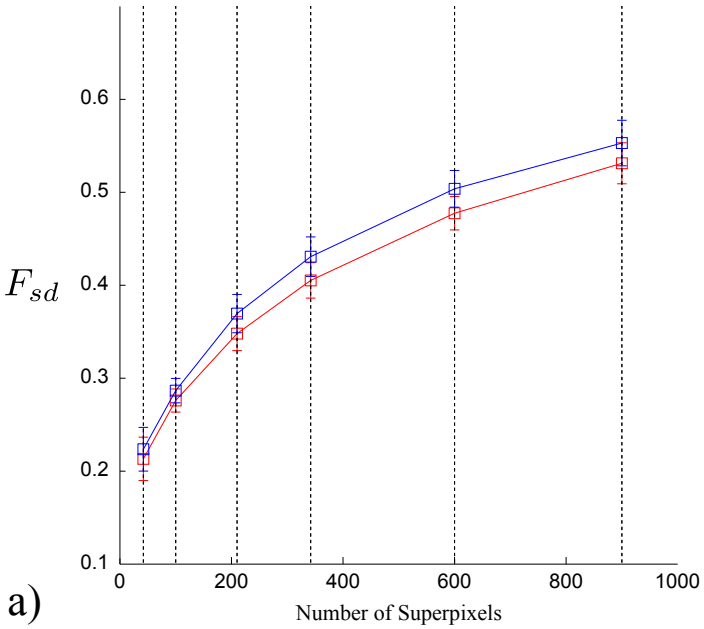


Figure 4.15: Foveation of warped lattices. a) Original image. b) Scene shape warped image. c) GRL algorithm based on original image without prior. d) GRL algorithm based on warped image. e) Un-warped superpixel based on the backwards warp of [d]. Notice the foveation of the lattice with small superpixel appearing in the center of the image.

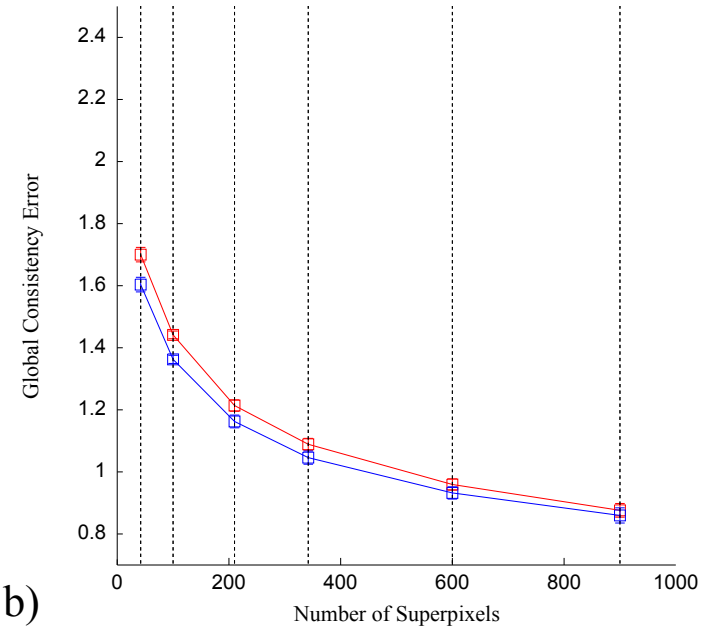
4.5.2 Comparison to GRL algorithm

To directly assess the effect of the learned spatial prior, we first compare our adaptive regular lattice algorithm (ARL) to the original greedy regular lattice algorithm (GRL) from the last chapter. An example of segmentations with and without the prior are shown in Figure 4.15. We can see that the prior causes the shape of the grid to be pulled towards regions of the image that are estimated to have the greatest probability of a boundary. This means there are more superpixels per unit area in regions that are likely to contain small objects. In Figure 4.16 we can see graphs showing the improved performance for the measures F_{sd} and *Global Consistency Error*.

An example of the intuition that the ARL algorithm better captures small objects is shown in Figure 4.17. Figure 4.17b shows that cars at a distance were missed using the GRL algorithm. However, using the ARL algorithm the foveation caused by the prior means that the cars are correctly detected, see Figure 4.17d.



a)



b)

KEY: GRL (BEL) ARL (BEL)

Figure 4.16: Comparing performance of GRL presented in Chapter 3 with ARL using the Scene Shape Prior. a) Number of superpixels vs F_{sd}. b) Number of superpixel vs Global Consistency Error. In general there is improvement across all classes. However analysis in Table 4.4 shows there is greatest improvement in classes that vary in size across images.

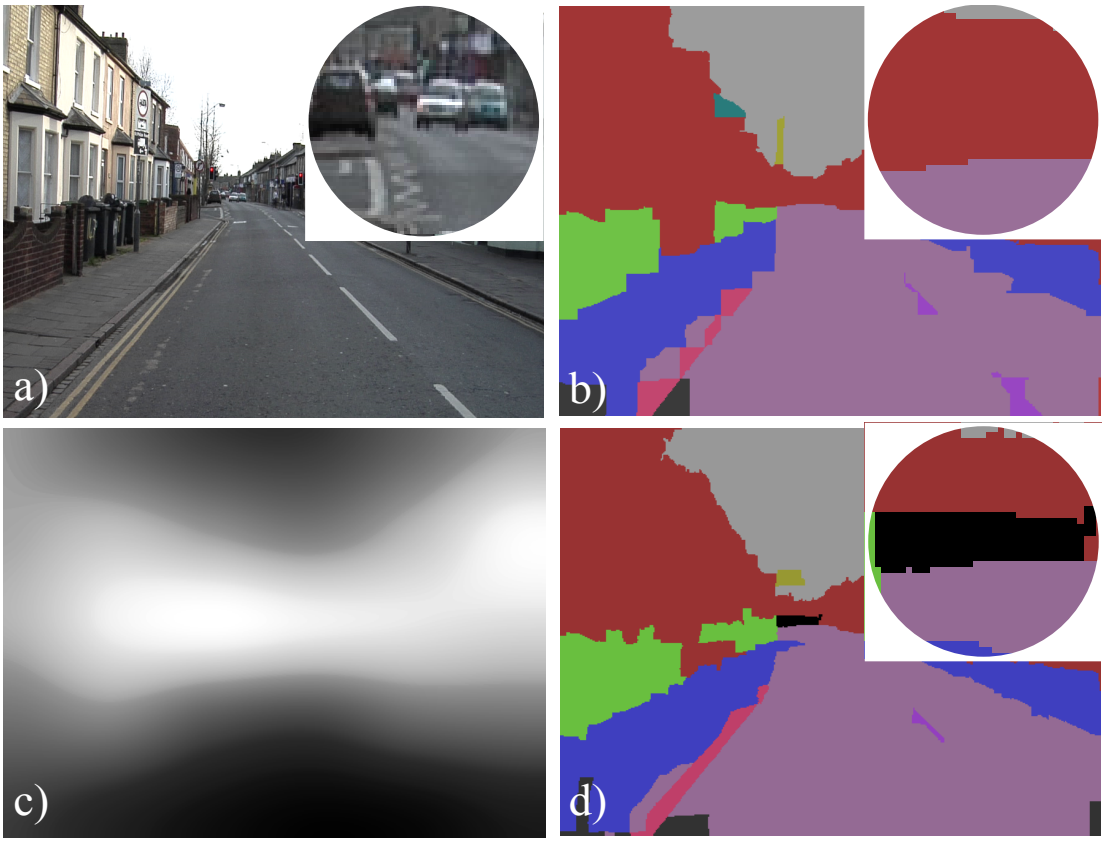


Figure 4.17: Scene shape prior helps capture small objects. a) Street scene. Inset: cars in the distance. b) Segmentation without boundary prior. Inset: Small distant objects (cars) missed. c) Learned boundary distribution prior. d) Foveated segmentation based on learned prior. Inset: improved segmentation of distant objects.

4.5.3 Comparison to other algorithms

In this section we compare the ARL algorithm to other competing methods. Table 4.2 shows the Borda scores for competing methods. The ARL-MS algorithm is ranked highest against competing methods. We can gain more insight by looking at the F_{sd} score for individual classes in Table 4.3. Here we can see that the ARL algorithm does particularly well on classes that vary in size in the image depending on the scene. For instance, the car class improves from a score of 0.63 with GRL to to 0.72 with ARL. Unsurprising the clustering algorithms, FH and MS, do best on homogeneous classes like Sky. However they also perform particularly well on very thin classes like Column/Pole. A second interesting thing to notice about the FH algorithm is that it performs well for detection but less well for segmentation, which can be seen by comparing the F-scores in Figure 4.18. This may suggest why the algorithm is visually appealing but performs less well quantitatively overall.

Considering again the Column/Pole class there is a drop from a score of 0.19 using the GRL algorithm to 0.18 using the ARL algorithm which suggests that the prior is making segmenting this class harder. This might be because the prior can warp long thin objects differently over parts of the image making them harder to follow with single paths. An example of this can be seen in the top of Figure 4.15b.

However, if we consider mean segmentation and detection in Figure 4.18 ARL-[MS] algorithm performs best. For instance, we can see from the box-whisker plot in Figure 4.19a that there are no poor performing outliers.

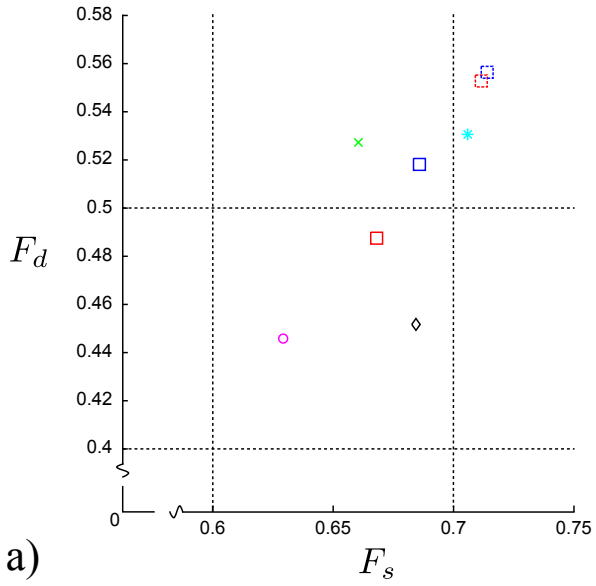
We perform statistical tests on the distribution of each segmentation algorithm using the functions anova1.m and multcompare.m. This performs a multiple comparison test using a 'Tukey-Kramer' correction for the significant difference criterion. This analysis is shown in Figure 4.19b. The ARL-[MS] algorithm is a significant improvement over most competing methods. In general the ARL algorithm has the advantage that it uses larger superpixels in the regions dominated by sky and road regions and can therefore model other areas of the image in greater granularity. This can be seen in Figure 4.20 where large regions of the road are captured by single superpixels. However, comparing the GRL and ARL algorithms with merging is interesting because the advantage of having large superpixels in certain regions is achieved through two mechanisms: warping and merging. From Table 4.4 we can see that while the overall performance between GRL-[MS] and ARL-[MS] is very similar (and GRL-[MS] sometimes has an advantage on the F_{sd} measure) the ARL-[MS] algorithm tends to improve performance on classes that vary more with perspective.

4.6 Discussion and Future Work

In the previous sections we have demonstrated that it is possible to learn a model of the spatial distribution of boundaries in an image and shown one possible way this can be used to improve segmentation performance. One way of evaluating the performance is to redraw the ground truth data, in the manner seen in Figure 4.3, using the learned warp on the ground truth data. This can be seen in Figure 4.21, where we note the more homogeneous size and distribution of ground truth objects in the warped images. However, there is considerable scope for future work, both in developing the model we have introduced here and for investigating alternative approaches.

Algorithm	Reference	Dataset
		CamVid [39]
ARL (BEL)	[183] ([74])	251
ARL-[MS] (BEL)	[183] [[53]] ([74])	**274**
FH	[88]	159
GRL (BEL)	[182] ([74])	236
GRL-[MS] (BEL)	[182] [[53]] ([74])	271
MS	[53]	167
NC (Pb)	[186] ([174])	269
uniform		180

Table 4.2: Algorithm ranking using Borda score on CamVid dataset. ARL improves performance over GRL. ARL-[MS] produces best overall performance. Maximum Borda score on individual dataset = 288.



KEY:

□ ARL, ⬚ ARL-[MS], × FH , □ GRL, ⬚ GRL-[MS], ∗ MS, ◊ NC, ○ Uniform

Figure 4.18: Plots of mean F_s against mean F_d for $\sim$ 600 superpixels. Note the improvement in performance of ARL over GRL. The FH algorithm also performs better at detection that segmentation compared to competing methods.

Algorithm	Classes										
	Bicyclist	Building	Car	Column/Pole	Fence	Pedestrian	Road	Sidewalk	Sign/Symbol	Sky	Tree
ARL (BEL)	0.380	0.612	0.721	0.182	0.729	0.381	0.592	0.805	0.354	0.672	0.566
ARL-[MS] (BEL)	**0.473**	0.636	**0.767**	0.233	**0.756**	0.456	0.601	**0.817**	0.456	0.681	0.579
FH	0.296	0.559	0.580	0.365	0.479	0.278	**0.614**	0.592	0.496	**0.803**	0.471
GRL (BEL)	0.338	0.614	0.627	0.193	0.697	0.316	0.589	0.774	0.293	0.680	0.553
GRL-[MS] (BEL)	0.456	**0.637**	0.723	0.269	0.740	0.439	0.597	0.789	0.434	0.691	**0.588**
MS	0.333	0.613	0.643	**0.414**	0.595	**0.564**	0.566	0.626	**0.533**	0.736	0.425
NC (Pb)	0.322	0.603	0.608	0.158	0.651	0.373	0.563	0.707	0.337	0.669	0.556
uniform	0.287	0.608	0.549	0.058	0.661	0.293	0.559	0.754	0.259	0.648	0.545

Table 4.3: Results of F_{sd} by class for competing algorithms at ~ 600 superpixels. Performance on classes that very in size with perspective show considerable improvement with the ARL algorithm. For instance, pedestrian and cars classes show improved performance of 7% and 9% improvement from GRL to ARL.

Algorithm	GRL-[MS] (BEL)						ARL-[MS] (BEL)					
Approximate Lattice Resolution / Selected Classes	~42	~100	~210	~342	~600	~900	~42	~100	~210	~342	~600	~900
	6 × 7	10 × 10	14 × 15	18 × 19	24 × 25	30 × 30	6 × 7	10 × 10	14 × 15	18 × 19	24 × 25	30 × 30
All Classes	0.297	**0.379**	**0.464**	**0.525**	0.579	0.591	**0.299**	0.373	0.460	0.518	**0.581**	**0.625**
Car	0.197	0.386	0.512	0.647	0.722	0.749	**0.220**	**0.411**	**0.575**	**0.698**	**0.767**	**0.796**
Pedestrian	0.048	0.142	0.227	**0.355**	0.440	0.459	**0.053**	**0.145**	**0.234**	0.344	**0.456**	**0.503**
Sign/Symbol	0.025	**0.092**	0.176	0.307	0.434	0.472	**0.031**	0.085	**0.177**	**0.313**	**0.456**	**0.532**

Table 4.4: GRL-[MS] vs ARL-[MS] for selected CamVid classes using F_{sd}. The performance for both merged lattices is similar but those objects that have greater variation in size as a result of perspective effects are better captured by the ARL algorithm.

4.6.1 Comparison to existing methods

We approached the problem of the uneven distribution of objects within a scene by learning a prior for the segmentation from real-world boundaries in training data. An approach based on object boundaries was largely motivated by the *boundary cost map* that forms the basis of the `GRL` algorithm. However, we note that other methods would be possible. For example, Hoiem et al. [119] estimated surface geometry and camera viewpoint to determine a prior over object sizes and a similar approach could be used to influence segmentation. Essentially, our model has learnt that road scenes tend to be foveated without high level information about the perspective projection process.

Additionally, the use of low resolution priors for guiding segmentation also has interesting analogies to other work. For instance, the title "scene shape" was motivated by the notions of "shape envelope" [198] used for scene recognition. While our method is based on a local classifier (and therefore not directly comparable to the use of global properties of the scene to influence further vision tasks) the information in the image that is exploited is similar. However, it is interesting to note that our results are achieved without any explicit object knowledge [31], object location priors [260, 115], or class specific edges [74, 208].

One natural question to ask is whether different segmentations should be provided for different classes in a scene. This would extend work on multiple segmentations [172, 138] based on hypotheses about which objects are in the scene and where they are likely to occur spatially [260]. For instance, the poor performance of all algorithms

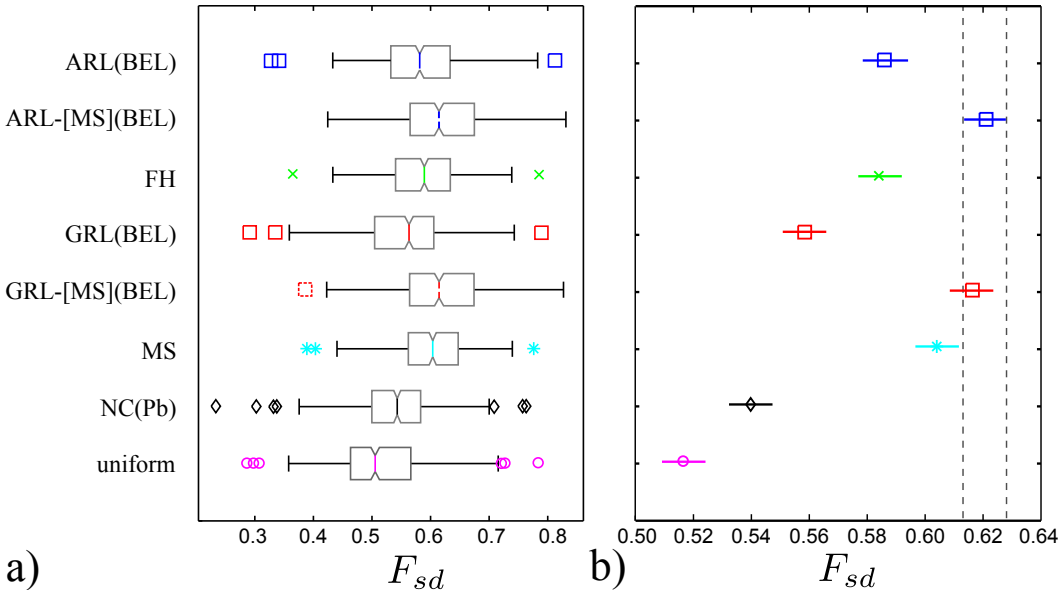


Figure 4.19: Statistical comparison of methods for ~ 600 superpixels. a) Box-whisker plots using F_{sd}. ARL-[MS] performs best overall and has no poor performance outliers. Outliers are marked by marker to left and right of whiskers. b) Plot of means and standard error for competing methods. The analysis demonstrates that the ARL-[MS] produces a significant advantage over most competing methods.

on the Column/Pole class suggests that an independent segmentation with the sole goal of identifying these thin structures might have better success. Poor recognition performance resulting from bad segmentations for this class has been noted independently in the work of Sturgess et al. [248] and one solution is to use specific object detector in an integrated framework [149].

4.6.2 Evaluation

Our evaluation can be extended in several directions. First, setting parameters for different components of the model could be investigated using a validation set. Further experiments should be used to gauge the influence of the number of factors/clusters and some analysis of the sensitivity to initial conditions during learning and inference. Results in Table 4.1 and Appendix B show that there is a modest advantage of using the clustered model over a single factor matrix. However, the partition of the data proposed by Brostow et al. [40] means that the driving manoeuvres seem less varied in the test set that training sets. We note that in the experiments 75% of test set examples are drawn from one cluster of our learned model. The true benefit of using the cluster model may

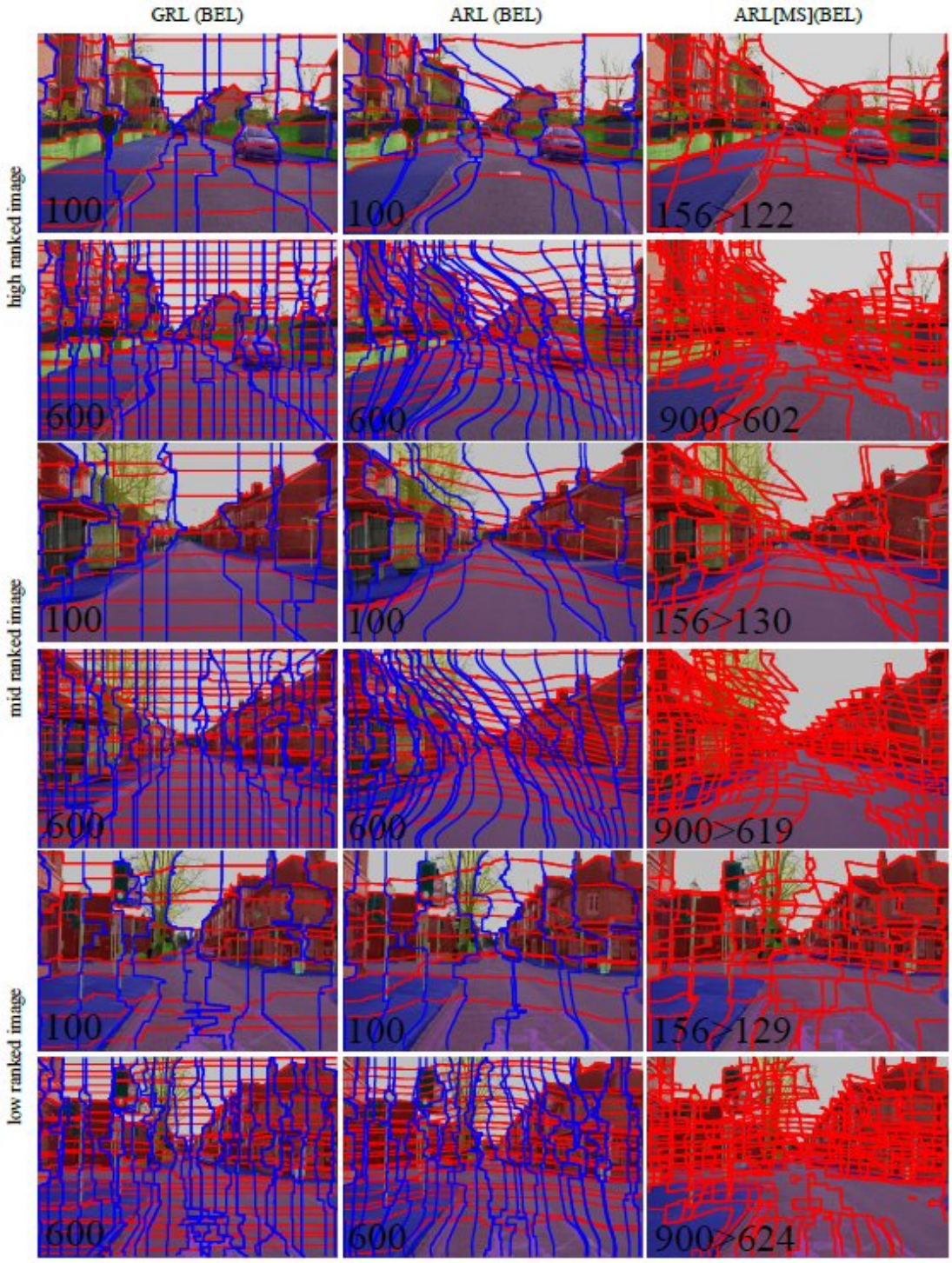


Figure 4.20: High/Mid/Low ranked images from CamVid comparing GRL and ARL. Note the foveation that occurs in the ARL algorithm. Results may be compared to those in Figure 3.25 and 3.26. Bold numbers which of GRL-[MS] or ARL-[MS] algorithms is dominant at a particular lattice resolution on object class.

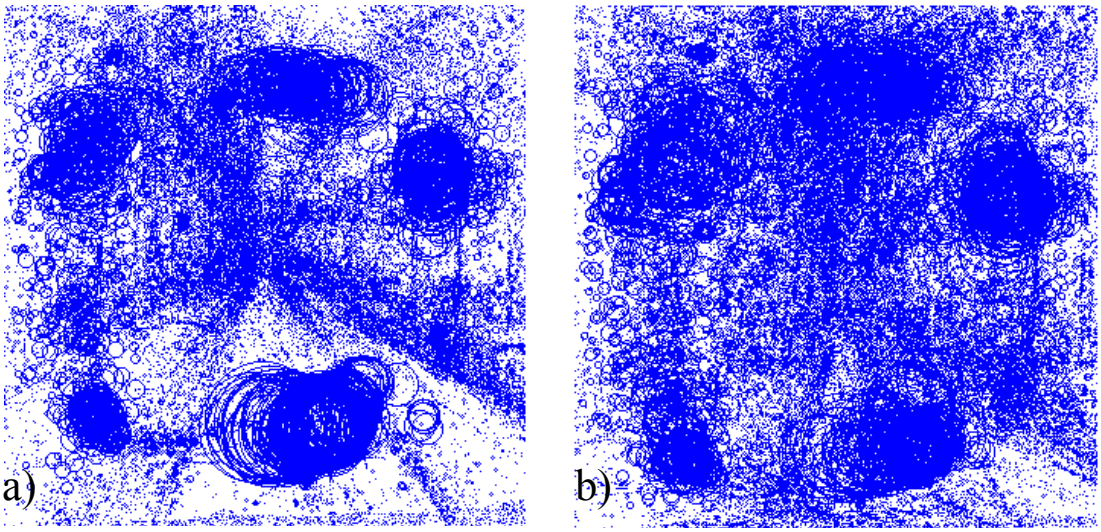


Figure 4.21: Plots to show the size and distribution of CamVid [39] dataset ground truth regions. a) Unwarped images. b) Warped images. Note the more homogeneous size and distribution of objects in the warped images.

be underestimated using this partition of the data. Cross-validation under different partitions would be a useful exercise for not only for gauging the robustness of our model but for highlighting peculiarities of the capture conditions between different partitions.

Second, it should be noted that the warps generated by two methods in Sections 4.3.1-4.3.2 and Section 4.3.3 are not identical. The results reported in a previous publication of this work [183] used the method of warping each strip individually and this is the fastest approach for a practical application. The warp also separates the vertical or horizontal warps which means the parameters could be adjusted separately. However, the results presented in this chapter were achieved using the *whole-image* warp (Section 4.3.3). This approach was more efficient over sets of images because we could use our fast publicly available code for the `GRL` algorithm. The experiments carried out in this chapter supports the same conclusions to those drawn in the published paper but a detailed empirical analysis of the two approaches is left for future work. Importantly, the *whole-image* warping offers a set of techniques that can alter the behaviour of other existing segmentation algorithms. For instance, superpixel algorithms with a compactness constraint, in this case `Uniform` and `NC` will also produce foveated segmentations if run on the warped image. We have seen an improvement in performance on selected images using both these algorithms and we shall also make use of the method in the

work presented in the next chapter.

4.6.3 Limitations of the algorithm

We have only used the BDI to steer a greedy regular lattice towards regions of the image where we think there are object boundaries. It would be possible to adopt the opposite approach and use the same technique to steer the lattice away from regions that are hypothesized to contain whole objects. This would mean that the modeling power of the lattice is concentrated in as yet uncertain regions of the image and would present one possible way for the different tasks of detection, recognition and segmentation to influence each other [120]. The relationship between steered segmentation (using the BDI) and the tasks of saliency and visual search could be investigated, where alternative measures are exploited to direct the algorithm [273, 103]. This direction of research may lead towards a sampling theory of segmentation.

It is unclear how the CLT model would work on alternative datasets (for instance BSD and VOC) or how the technique would suffer with very low levels of training data. One possibility would be to use semi-supervised techniques for learning when the availability of human labeled data is limited but algorithmic data (the output of boundary detectors) is high.

Lastly, the temporal properties of the prior have not been explored in this work. There is no guarantee that temporal transitions over a sequence of images using a clustered model would be smooth and this may be useful in certain applications.

4.7 Conclusions

The use of a novel prior to influence the distribution of strips within the image is one approach to help improve our `GRL` algorithm, by not restricting each path to a fixed region of the image. However, the underlying method is still a greedy solution and it is likely to produce sub-optimal performance.

Additionally, the analysis in this chapter reveals that despite our improved performance some object classes have F_{sd} scores ~ 0.20 with even large numbers of superpixels. While it remains unclear how far it is possible to get with segmentation alone, rather than incorporation detection or recognition information [149], this low performance suggests there is still plenty of scope for producing better algorithms.

In the next chapter we introduce a new algorithm, and a different graph construction, that allows us to construct a lattice that does not use greedily selected fixed strips within an image.

Chapter 5

Lattice Cut

In this chapter we recast the superpixel lattice problem, introduced in Chapter 3, as a Markov random field. We first show how to construct a graph in which the MAP labelling is guaranteed to be set of labels arranged in ordered layers. We then introduce an algorithm that uses a coupled MRF formulation to learn colour distributions in an iterative manner across the image to produce a final lattice. We demonstrate improved performance over algorithms presented in previous chapters.

5.1 Introduction

We begin this chapter with a review of one approach to supervised segmentation based on a Markov random field model, *GrabCut* [225], which is a simple and elegant algorithm for foreground/background segmentation that motivates the contributions in this chapter.

Classical approaches to supervised segmentation problems use either texture or colour information, eg. Magic Wand [Adobe Systems Incorp. 2002], or edge information eg. Intelligent Scissors [187]. By using a *Graph Cut* approach to the segmentation problem the *GrabCut* framework can incorporate both region and boundary information.

The algorithm proceeds in four steps: 1) Assign mixture model components to pixels. 2) Learn mixture model parameters. 3) Estimate current binary labelling using *GraphCut* (see Figure 2.7b). 4) Repeat from Step 1 until convergence.

The implementation presented in [225] initializes the background labels using a strip of pixels around the outside of the marked rectangle (see Figure 5.1a). The structure of the algorithm guarantees that the iterative method converges because each of

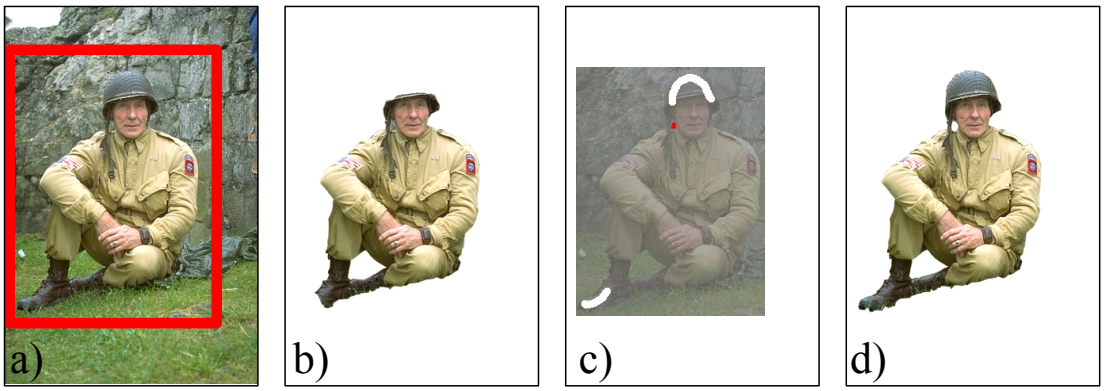


Figure 5.1: GrabCut. a) Original image and initial user bounding box. b) Supervised segmentation based on background model from initial bounding box. c) Additional user inputs for background (red) and foreground (white). d) Final segmentation based on iterative optimization scheme. Reproduced from [225].

the steps 1 to 3 can be shown to be a minimization of the total energy with respect to the full set of parameters. The *GrabCut* algorithm produces convincing performance on real world examples.

In contrast to the *GrabCut* approach the superpixel algorithms we have encountered in previous chapters have a common weakness that they *either* use pixel color statistics [53, 88] *or* are based on a pre-computed boundary map [182, 186].

A second problem is that most methods rely on greedy [182] or approximate optimization strategies [238]. Algorithms that do find a globally optimal solution, eg. [88], have cost functions that tend to lack stability such that a small change in the image can have a dramatic effect on the overall segmentation (see Section 3.10). Again, this is in contrast with the *GrabCut* approach. For example, although *GrabCut* [225] does not guarantee an optimal solution without known colour models, it is based on an alternating optimization scheme in which a global optimum is found at each iteration *given fixed color models* and then the color models are updated. Interestingly, this process of iteratively estimating properties of a new image to update a superpixel segmentation is also absent in the methods we have encountered so far.

In this chapter we draw on work from supervised segmentation to help improve upon these deficiencies. We have the following desiderata:

1. Introduce costs based on the superpixel region of support, not just the boundaries between them.

2. Estimate, iteratively, properties of pixels within a superpixel in a novel image.

3. Produce global guarantees on the spatial distribution of superpixels over the whole image.

To achieve this we make use of recent work on Markov random fields [128, 231]. The algorithm we introduce exploits both region and boundary information and is based on an alternating optimization strategy.

In Section 5.2 we first give an overview of our proposed method. We show that for it to work we need to place certain restrictions on a set of labels that correspond to different sets of superpixels. Then in Section 5.3 we review work on MRFs and show how to impose restrictions on the label field. The full algorithm and a discussion of the parameters follows in Section 5.5.

5.2 Overview of our approach

An overview of our approach can be seen illustrated in Figure 5.2. We begin with a regular grid of superpixels and then alternate between improving the vertical boundaries between superpixels (black lines in Figure 5.2a-d) and the horizontal ones (red lines in Figure 5.2a-d). At each step, we obtain a solution for all of the vertical (or horizontal) boundaries simultaneously. Since the method for producing vertical and horizontal boundaries differs only by rotation we shall use the vertical orientation in our description.

We associate a label with each pixel, which indicates which row or column a superpixel belongs to. For example in Figure 5.2a-d, superpixels 1-4 take label 0 (column 1), superpixels 5-8 take label 1 (column 2) and so on. The vertical boundary between superpixels 1-4 and superpixels 5-8 is implicitly defined by the change in pixel labels from 0 to 1. It is useful to think of each label as representing a different layer in a 3D graph and the superpixel boundaries as occurring at the steps between layers. An illustration of this can be seen in Figure 5.2e-f.

The unary potentials in our MRF formulation reflect the tendency of a pixel to take one label or another and are based on statistical color models defined within each superpixel. Pairwise potentials encourage neighboring pixels to take the same label and hence belong to the same layer. We use boundary information in the pair terms so that

layers are encouraged to align with object boundaries.

To create layers of labels as in Figure 5.3a there are several constraints that must be obeyed by the label field:

Constraint 1 As we move from top to bottom along any column, the labels may only increment or decrement by a single value at a time ie. the labels are ordered sequentially (see Figure 5.3a column y).

Constraint 2 As we move from left to right along any row, the labels must increase sequentially and monotonically (see Figure 5.3a row x).

Constraint 3 In maintaining Constraint 2 EVERY label in the set must appear in every row. (see Figure 5.3d row x).

Combined, these three constraints maintain ordered layers of labels. However, these constraints also impose a restriction on the shape of the layers. The boundary of each layer is forced to be a non-returning path. This is the same restriction the we met using the dynamic programming method that we used in Chapter 3 Section 3.2.3.

To see why this restriction is necessary let us compare the ordering of neighbouring pixels in Figure 5.3b-c. The vertical boundary in 5.3b that doubles back on itself cannot be distinguished by the local ordering of two pixels from the two separate regions in 5.3c. Therefore to maintain the topology of ordered vertical layers we must limit the label field to increase monotonically left to right.

We shall examine how this restriction limits performance in Section 5.9. In the following sections, we describe a graph cut formulation that can find the MAP solution to a multi-label MRF while obeying these constraints and hence optimize all of the vertical superpixel boundaries simultaneously.

5.3 Multi-label MRFs and Constraint Edges

In this section we first briefly review multi-label MRF problems and introduce the notion of *constraint* edges. In Chapter 2 we saw that binary MRFs have been exploited in several different computer vision problems. To recap, let $\mathcal{P}$ be a set of image pixels, y_p be a label assigned to pixel p from a finite label set $\mathcal{L}$, and $\mathbf{y}$ be a collection of all pixel/label assignments. There are two components in the standard graph-cut energy formulation:

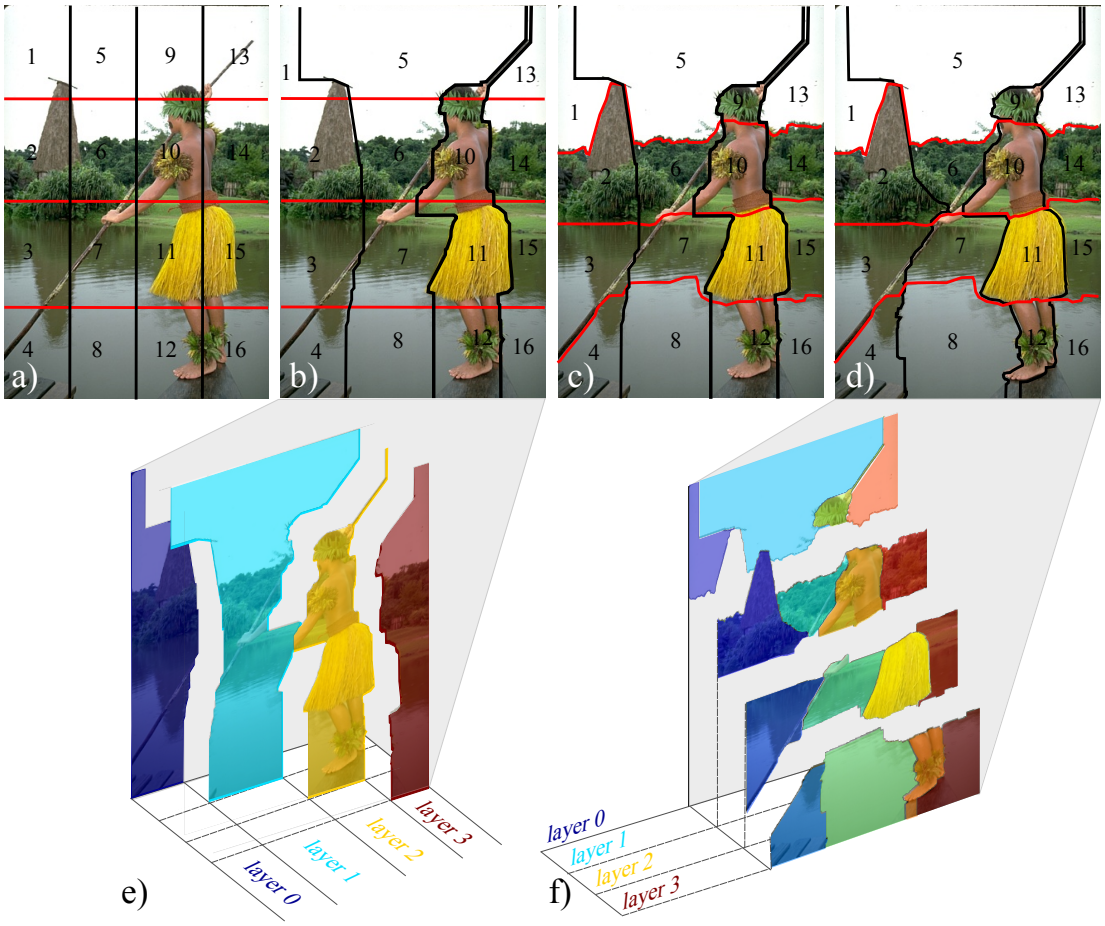


Figure 5.2: Lattice Cut. a-d) Starting with an evenly spaced grid of superpixels we alternately update the horizontal and vertical components of the lattice (red and black lines respectively) corresponding to the two sets of labels at each pixel — one for each horizontal or vertical layer. e) For the vertical update this label determines which of the vertical strips (or layers) the pixel belongs to. f) For the horizontal update, the label determines which of the horizontal strips the pixel belongs to. The cost function for each update depends on edges and the coherence of the resulting superpixel regions and the solution is guaranteed to maintain the lattice structure.

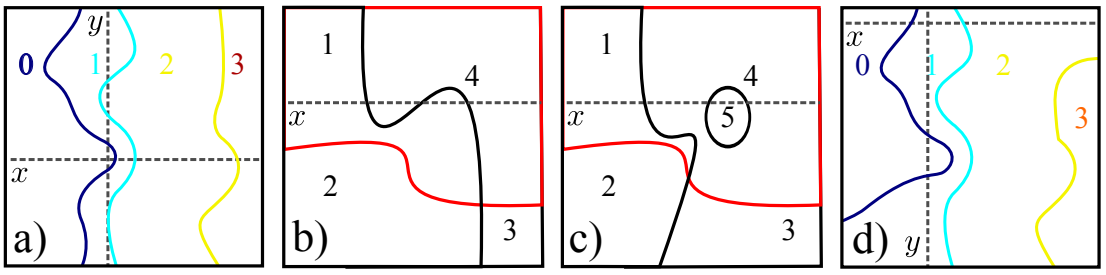


Figure 5.3: Label constraints. a) Constraint 1 requires that labels along any vertical path y are sequential (here 1-2-1-0-1). Constraint 2 requires that labels along any horizontal path x are monotonic sequential (here 0-1-2-3). b) Example of vertical returning path. c) Example of identical label ordering in row x to example [b] but where layer topology is violated. Comparing examples [b] and [c] shows why imposing **Constraint 2** is necessary for this graph construction. d) Lopsided layer solution that restricted by **Constraint 3**. Here only 3/4 of labels appear in solution on row x.

$$E(\mathbf{y}) = \sum_{p \in \mathcal{P}} U_p(y_p) + \sum_{p,q \in \mathcal{N}} P_{mn}(y_m, y_n) \tag{5.1}$$

The data terms U_p represent the cost for pixel p having label y_p and the regularization terms P_{mn}, defined on a neighborhood system $\mathcal{N}$, encourage spatial smoothness. Figure 2.7 shows the graph construction used to minimize the binary label problem. Recent work on multi-label MRFs has used novel graph constructions to find MAP solution using a single cut given an ordered set of labels with convex priors [128, 231].

We demonstrate the main ideas with a simple two pixel example. Consider two pixels at sites p and q, each of which can take one of three labels $y_p, y_q \in \{0, 1, 2\}$. The *equivalence* graph proposed by Ishikawa [128] for the multilabel energy has two columns of edges that represent the pixels p and q, see Figure 5.4a. For each column there are three downward pointing edges, each of which represents the unary potential for the three possible labels at that pixel. For example, the edge representing label 2 at pixel p is weighted by U_p^2 (the cost at pixel p for assigning label 2). If this edge is part of the final cut we pay this unary cost.

In a minimum cut that separates source from sink we must cut exactly one of the three downward links at each pixel and this choice determines the assigned label. To achieve this Ishikawa [128] proposes placing an edge of infinite cost between nodes $i + 1$ and i in each column. This makes each encoding of the label within each column

unique by forcing a cut that has more than one edge from each column to have an infinite energy, $E(\mathbf{y}) = \infty$. The proof can be found in [128].

Horizontal potentials $P_{pq}^{01}, P_{pq}^{10}, P_{pq}^{12}, P_{pq}^{21}$ are also shown in Figure 5.4a. Here the notation P_{pq}^{kl} denotes the pairwise cost for assigning pixel p to label k and pixel q to label l. If we assign both pixels to the same label these edges do not appear in the cut: we pay no penalty (i.e. the contribution of the pairwise terms P_{pq}^{kk} to the energy of the configuration is zero) and this encourages smoothness in the label field (Figures 5.4b-c). If the assigned labels differ by one, then we must cut exactly one of the horizontal links: there are four possible ways the labels can differ by one (01,10,12 and 21) and each has an associated horizontal edge and cost (examples in Figures 5.4d-e).

5.4 Graph construction for ordered layers

Unfortunately, this graph also assigns a finite cost for configurations where the labels differ by more than one. For example the solution where p takes label 2 and q takes label 0 (Figure 5.4f) incurs a pairwise cost of $P_{pq}^{21} + P_{pq}^{10}$. However, **Constraint 1** requires our labels to be sequential, so in the final graph it should *never* be possible to have this solution.

To achieve this we return to the idea of *constraint* edges. To prevent non sequential labelling we add diagonal edges $\{q_2, p_1\}$ and $\{p_2, q_1\}$ with infinite cost. Now any solution that assigns label 2 to pixel p and label 0 to pixel q must also cut the *constraint edge* $\{p_2, q_1\}$ and hence has an infinite cost. Similarly, the link $\{q_2, p_1\}$ prevents the solution where the pixel p takes label 0 and pixel q takes label 2. These sequential constraint edges are illustrated in Figure 5.5a. A configuration with infinite cost can be seen in Figure 5.5b.

We have now constructed a graph in which **Constraint 1**, that neighboring pixels must change sequentially is, enforced. Similarly **Constraint 2** ensures labels increase left to right. This can be achieved by making constraint edges of P_{pq}^{10} and P_{pq}^{21} (see Figure 5.5c). Again we add the edge $\{q_2, p_1\}$ so that the label increase is incremental. This constraint means that solutions in which a label decrements as we move from pixel p to pixel q have infinite cost. For example, in Figure 5.5b the solution $2 - 0$ is now also prevented by the infinite capacity of edge $\{p_2, q_2\}$.

Equivalently, we could leave P_{pq}^{10} and P_{pq}^{21} as they are and add infinitely expensive

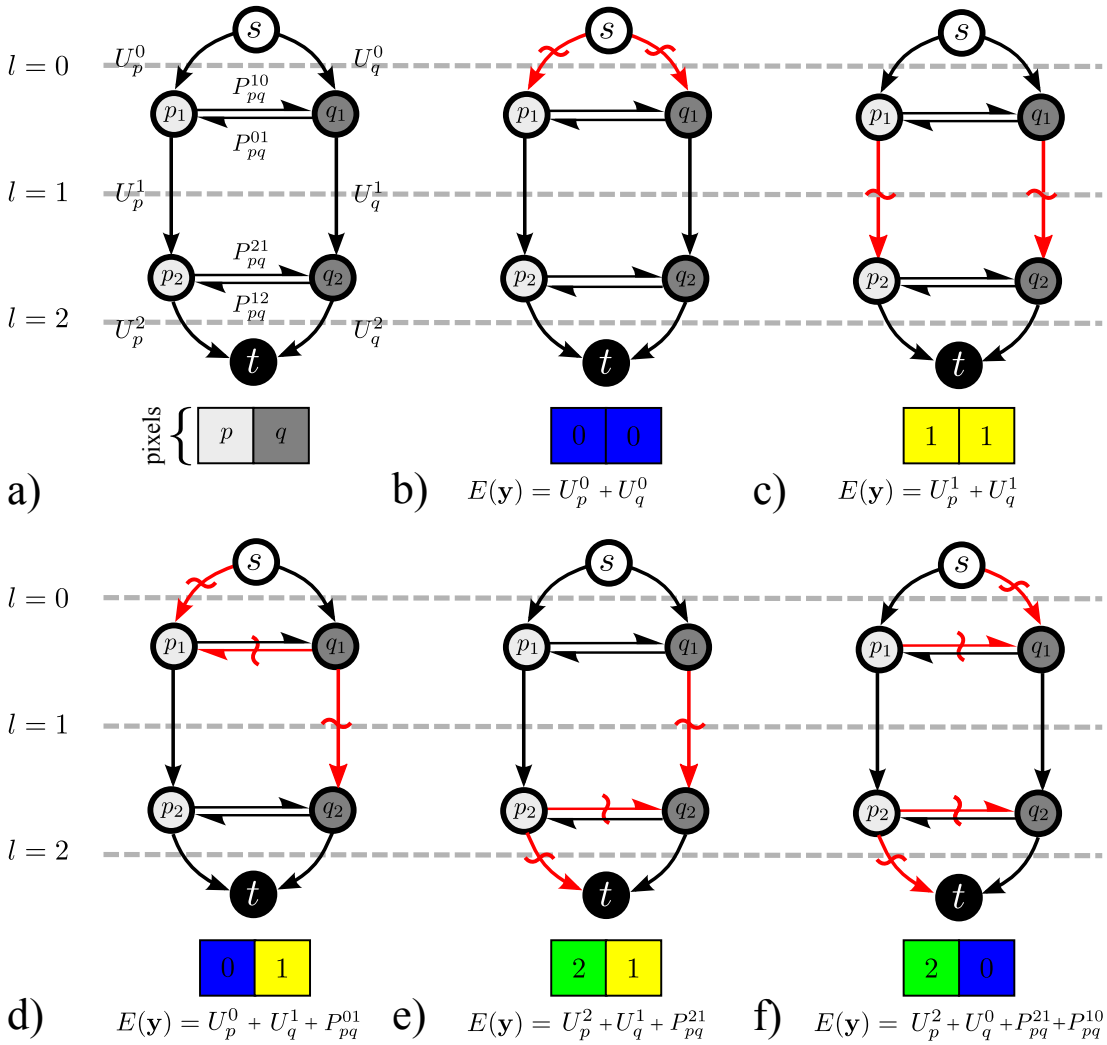


Figure 5.4: a) Example graph construction with two pixels p and q, each of which can take label $l \in \{0, 1, 2\}$. Unary costs are associated with vertical links. U_p^k is the unary cost for assigning the k'th label to pixel p. Pairwise costs are represented on horizontal links. P_{pq}^{mn} is the pairwise cost assigning the m'th label to pixel p and the n'th label to pixel q. The labelling is determined by which vertical links are cut. b) Topmost vertical links are cut and so both pixels are labelled as zero. c) Both pixels are labelled as 1. d-e) Cuts where the labels change sequentially also incur a pairwise cost. f) Unfortunately, cuts where labels change non-sequentially are also permitted in this construction: these solutions violate constraints 1 and 2 and should be suppressed (see Figure 5.5).

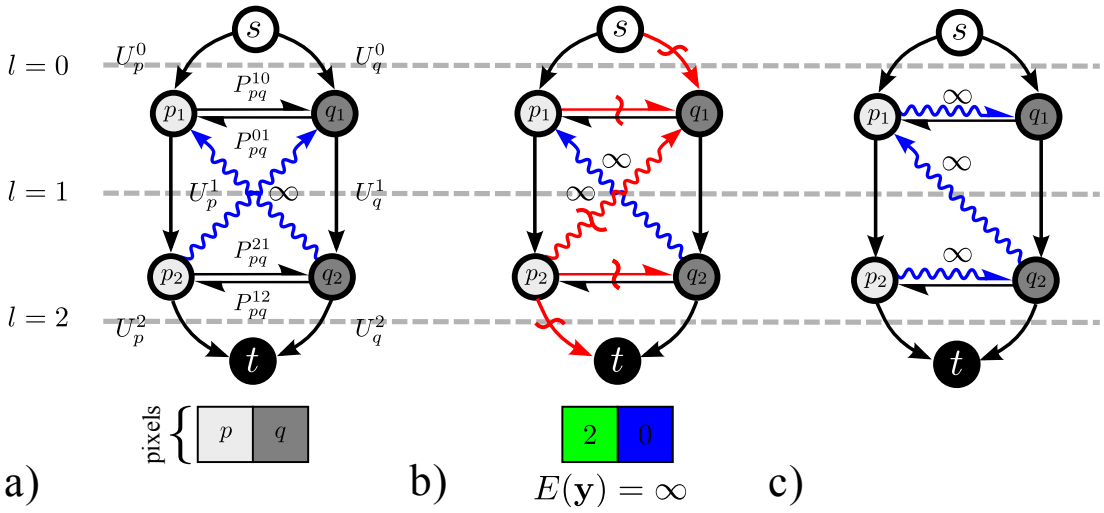


Figure 5.5: Imposing sequential constraints a) Diagonal constraint links with infinite cost prevent non-sequential labels (constraint 1). b) To label the pixels 2,0 as in Figure 5.4f we must now also cut one of these diagonal links: the cost for this solution becomes infinite and it will never be chosen. c) To additionally force the label map to sequentially increase (constraint 2) we replace the relevant pairwise terms P_{pq}^{10} and P_{pq}^{21} with constraint links of infinite cost. This gives labellings like that in Figure 5.4e an infinite cost.

constraint edges $\{p_1, q_1\}$ and $\{p_2, q_2\}$. The second interpretation, of two sets of edges in the graph is useful. The first set of edges encode the potentials we use to model the relationship between data and random variables in the label field. The second set of edges - the *constraint edges* - restrict the set of feasible solutions on the label field. This formulation is due to Delong and Boykov [67] and results in an additional pairwise term in our standard MRF cost function:

$$E(\mathbf{y}) = \sum_{p \in \mathcal{P}} U_p(\mathbf{y}_p) + \sum_{p,q \in \mathcal{N}} P_{pq}(\mathbf{y}_p, \mathbf{y}_q) + \sum_{p,q \in \mathcal{M}} Q_{pq}(\mathbf{x}_p, \mathbf{x}_q) \tag{5.2}$$

where Q represents the constraint terms that restrict the set of feasible solutions on the graph over a set of labels $\mathcal{L}$ with neighborhood system $\mathcal{M}$. The formulation helps to emphasise that the neighbourhoods for the potential and constraint terms need not be the same.

Lastly, **Constraint 3** can be imposed with a further set of constraint edges that force the labels at each end of the graph to take on labels for the first and last set of layers. This is illustrated in Figure 5.6.

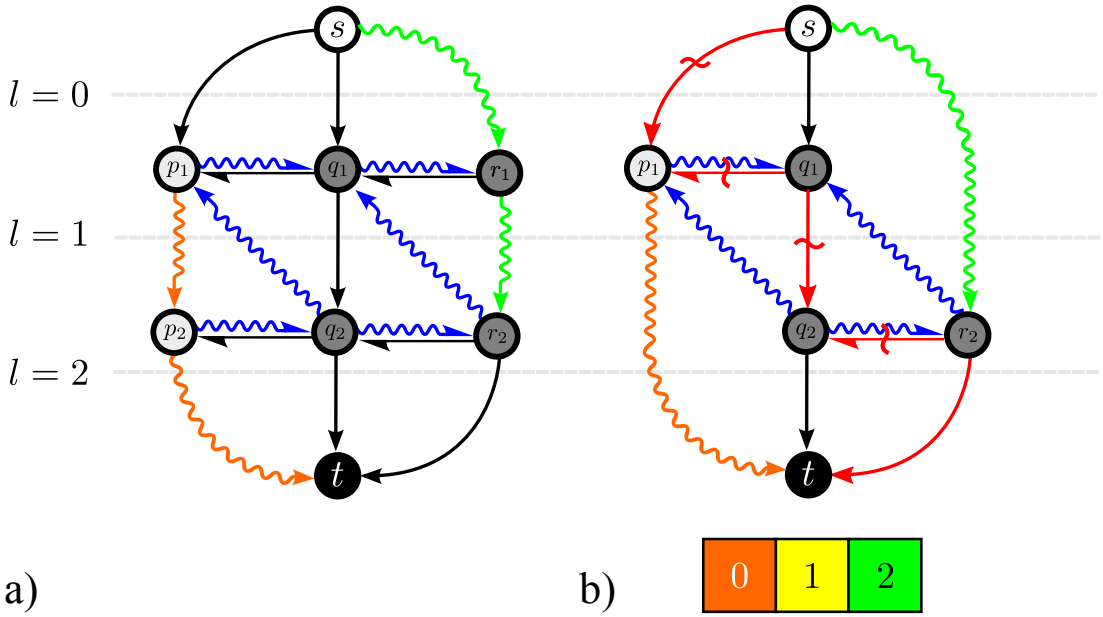


Figure 5.6: Enforcing all layers. a) To ensure each layer appears in the solution (**Constraint 3**) we set constraint edges between labels that cannot appear in the cut. In this 3 pixel example this corresponds to the colored edges in first and last column. b) Edges and nodes that encode redundant labels can be removed from the graph to reduce memory requirements ie. node p_2 and r_1. The only feasible labelling on this graph is 0-1-2. Constraint edges that force a particular label to appear in the solution are colored according to the label.

5.4.1 Full graph construction

The full graph construction to guarantee a solution as in Figure 5.3a is as follows. For an image with $X \times Y$ pixels and $|\mathcal{L}|$ possible labels, we build a graph with $|\mathcal{L}| - 1$ layers of $X \times Y$ nodes (Figure 5.7a). The topmost layer connects to the source and the bottom most layer connects to the sink. Viewed from above, the final cut on this graph will look like a set of stepped layers very much like Figure 5.2e.

Horizontal constraint edges between nodes at $\{x, y, l\}$ and $\{x+1, y, l\}$ ensure that the labels increase monotonically as we move from left to right. Diagonal constraint links with infinite capacity between the node at $\{x, y, l\}$ and $\{x, y-1, l-1\}$, $\{x, y+1, l-1\}$, $\{x-1, y, l-1\}$ and $\{x+1, y, l-1\}$ prevent non-sequential labels at neighboring pixels[1].

The diagonal constraint links effectively force the minimum width of each super-

[1]These additional edges preserve order in 4-connected sense and are readily extended to higher conductivities. Note that neighbourhoods $\mathcal{N}$ and $\mathcal{M}$ do not need to be the same.

pixel to be one. In fact it is possible to include additional diagonal constraint links to all nodes in the layer above within some radius. This has the effect of constraining the width of each superpixel to be at least equal to this radius but increases the connectivity and therefore memory requirements and runtime of the st-MINCUT algorithm.

Finally, we would like the superpixels to be *compact* [159]: the vertical bands should to be distributed roughly evenly over the image and the superpixels limited in size. This can be ensured by limiting the possible labels to a subset $\mathcal{K} \subset \mathcal{L}$ at each horizontal position so that at the left of the image only the first few labels can be chosen and at the right only the last few. This is illustrated in Figure 5.7b. To this end, we prune the nodes from the graph that represents undesirable combinations of pixels and labels. The result, a "sliced layered cake", appears as a diagonal strip in cross-section. The pruning also has the desirable effect of reducing the computational cost and storage requirements of the algorithm.

The full graph construction can minimize an energy exactly in polynomial time by finding the st-MINCUT of the graph. We shall refer to this algorithm for solving a multi-label problem that enforces a set of layers as *LayerCut*. In the following section we shall show how we can use this to produce a *superpixel lattice* algorithm that can be optimized in an iterative manner.

However, we first illustrate the use of *LayerCut* with a toy example where the ground truth label field obeys **Constraints 1-3**. In Figure 5.8a we can see a label field arranged in layers that might represent a tumor in a medical image or landscape feature in an aerial photograph. We simulate measurements from this ground truth data by drawing samples from different known Gaussian distributions for each label. This produces a set of layers that are very noisy and barely distinguishable, see Figure 5.8b. The difficulty of the labelling task is illustrated by showing the result based on the unary terms only, Figure 5.8c, which are very noisy.

If we apply a standard off-the-shelf algorithm for multi-label energy minimization such as α-expansion, Figure 5.8d, the solution to the label field improves but remains noisy. However, the solution using *LayerCut* improves on the result. Unlike α-expansion the *LayerCut* formulation imposes the required constraints in the label field and is also guaranteed to find the global optimum. Figure 5.8e illustrates a partial layer solution where only label order is constrained and the full solution is shown in

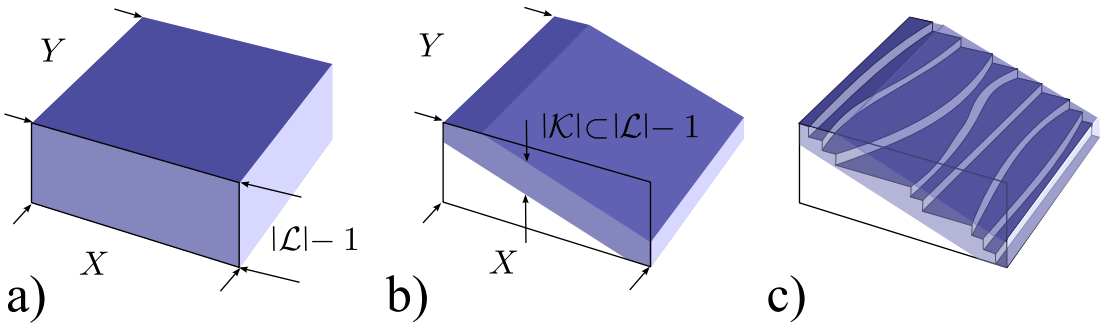


Figure 5.7: Full graph construction. a) For an $X \times Y$ image with $|\mathcal{L}|$ labels, we construct a graph with $X \times Y \times (|\mathcal{L}|-1)$ nodes. b) Layered cake construction. We limit the range of labels that can be assigned at any position X by removing nodes. This encourages regular superpixel spacing and reduces computation. c) Example cut produces a staircase with a step for each label.

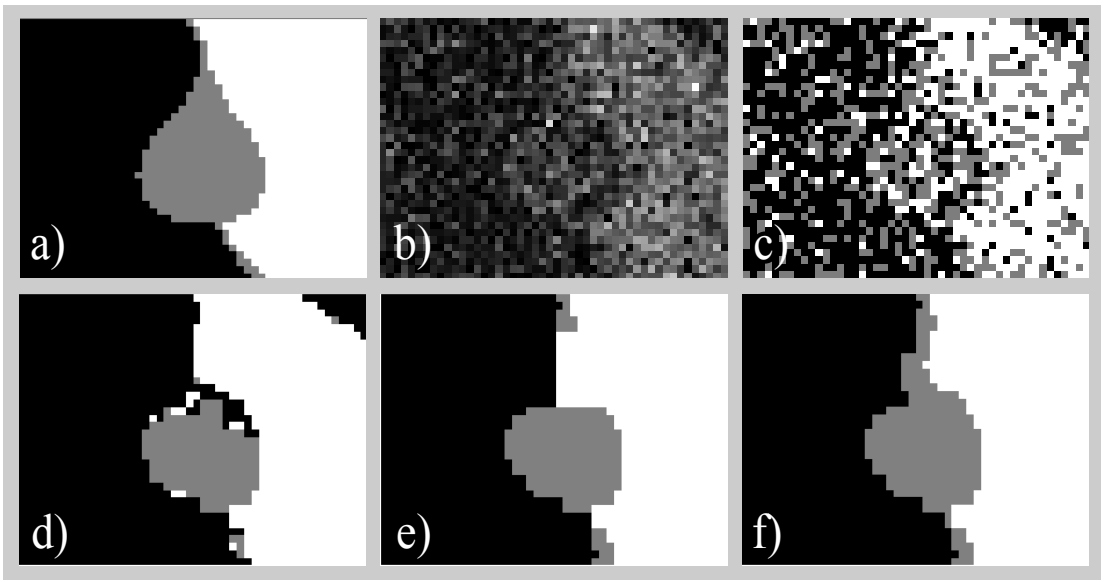


Figure 5.8: Toy example. River-Lake-Dam-River $|\mathcal{L}| = 3$. a) Ground truth labels. b) Observed data. c) Unary solution. d) α expansion [38]. e) Solutions constrained to increase. f) Solution constrained to monotonically increase. Note the thin grey label is ensured to appear left to right between black and white labels.

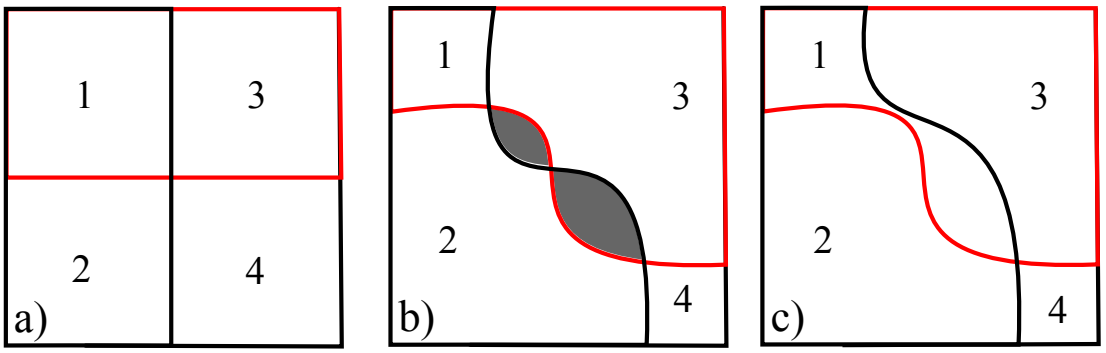


Figure 5.9: Lattice constraints on orthogonal MRFs. a) 2×2 lattice made up of 2 layer in the vertical (black) and horizontal (red) orientations. b) Non-lattice where the layer boundaries in orthogonal MRFs are not constrained to limit the number of crossing points. Labeling of gray regions show is ambiguous. c) A lattice with the same topology as [a] with only one crossing point.

Figure 5.8f. Note the absence of the black region in the top right hand corner which is present in the α-expansion solution.

5.5 Lattice Cut

Having produced a graph where the MAP solution is a set of layers we are able to a produce a superpixel lattice algorithm - *LatticeCut* - that is optimal in either the horizonal or vertical orientation. We use two MRFs in orthogonal orientations to solve for layers in the horizontal or vertical direction.

In addition to the three constraints in section 5.2 we must now also impose a fourth constraint that layers in orthogonal MRFs do not cross multiple times:

Constraint 4 Each vertical boundary must not cross each horizontal boundary more than once.

This is similar to the constraint that we introduced in Chapter 3. To impose the constraint we employ a similar technique where the set of edges along the label boundary in the orthogonal MRF are given a large fixed cost. Imposing this constraint is critical to maintaining the lattice structure of the resulting superpixels, illustrated in Figure 5.9. In this example the cost of of intersecting paths in Figure 5.9b is at least three times greater than the same paths in Figure 5.9c, due to the number of crossing points, and will not appear in the st-MINCUT solution.

Algorithm 4 Lattice Cut

$\mathcal{P} \leftarrow$ image, $\mathcal{B} \leftarrow$ [43, 74, 174] *//Set pixel and boundary data*

$\mathcal{S} = \{1, 2 \dots M \times N\} \leftarrow$ Uniform labelling *//Initialize superpixel regions*

$E(\hat{\mathbf{y}}) \leftarrow U + P, E(\mathbf{y}) \leftarrow \infty, iter \leftarrow 0$ *//Initialize energy and estimate*

//While combined cut is decreasing

while $E(\hat{\mathbf{y}}) < E(\mathbf{y})$ **do**

  //Alternately update estimate of superpixel regions and energy for each orientation

  $E(\mathbf{y}) \leftarrow E(\hat{\mathbf{y}})$

  if $mod(iter) = 2$ **then**

    $[\mathcal{S}, E(\hat{\mathbf{y}})] \leftarrow$ LayerCut($\mathcal{P}, \mathcal{B}, \mathcal{S}, M$) *//Do Horizontal LayerCut*

  else

    $[\mathcal{S}, E(\hat{\mathbf{y}})] \leftarrow$ LayerCut($\mathcal{P}, \mathcal{B}, \mathcal{S}, N$) *//Do Vertical LayerCut*

  end if $iter++$

end while

return $\mathcal{S}$

For a set of $M \times N = |\mathcal{S}|$ superpixels, pixel data $\mathcal{P}$ and boundary cost map $\mathcal{B}$ the *LatticeCut* algorithm proceeds as follows: We initialize a set of superpixels $\mathcal{S}$ and learn a set of models μ, one for each superpixel. Details of setting the potential terms are given in Section 5.5.2. We then solve the *LayerCut* problem alternating between orientations to iteratively update both the superpixel models and the region of support for each superpixel. At each iteration we update **Constraint 4** for the next graph based on the *LayerCut* solution to the previous iteration. The full *LatticeCut* algorithm is summarized in Algorithms 4 and 5.

Lastly, to complete our description of the algorithm we must specify a spatial arrangement of potentials, a schedule, for assigning unary potentials from each superpixel to layers in the full graph.

5.5.1 Model Grid

For our two MRFs to interact over a common set of superpixels we must assign a unary potential from a probabilistic model for each superpixel to a region of each layer in our full graph construction. To do this we assign the negative log likelihood to edges within

Algorithm 5 Layer Cut

1: **for all** $i \in \mathcal{S}$ **do**
2: *//Fit colour model to current superpixel data*
$\mu_i \leftarrow$ argmax $log(Pr(\mathcal{P}_i|\mu_i)Pr(\mu_i))$
3: **end for**
4: $U \leftarrow -log(Pr(\mathcal{P}, \mu))$ *//Set Unary terms* - Figure 5.10
5: $P \leftarrow \mathcal{B} \cap$ **Constraint 4** *//Set Pair terms*
6: $(\mathcal{S}, E(\hat{\mathbf{x}})) \leftarrow maxflow(U, P, Q)$ *//Do st-MINCUT*[35]
7: **return** $\mathcal{S}, E(\hat{\mathbf{x}})$

the region of support of each superpixel for every layer such that:

$$\mathbf{U}^l_{(i,j)} = -\log(\Pr(\mathcal{P}_j|\mu_i)) \tag{5.3}$$

where $\mathbf{U}^l_{(i,j)}$ indicates the unary potentials for label l and the parenthesis (i, j) indicates superpixel model μ_i for superpixel i and data $\mathcal{P}_j$ for superpixel j.

The unary potential schedule for the simple 2×2 lattice in Figure 5.9 can be seen in Figure 5.10. For example, if we consider the unary layer $\mathbf{U}^0$ for a vertical cut then we compare the data in superpixels 1 and 3 to our model of superpixel 1 and we compare the data in superpixels 2 and 4 to our model of superpixel 2. This way we compare the data in a region of support specific to each superpixel to competing superpixel models in neighbouring layers. Layers for each potential are not complete because they are limited by *compactness* of the full graph construction. Therefore the layers in Figure 5.10 correspond to the shape of the graph in Figure 5.7a rather than the "sliced layered cake" of 5.7b.

5.5.2 Parameters

In practice the models we use for each superpixel are not restricted by the *LatticeCut* formulation and could contain high-order cues like class labels or texture information [240]. However, we limit ourselves to simple colour models for the pixels within each superpixel which have been shown to work well in supervised applications [29]. We use multinomial distributions across the superpixel histograms in RGB space. We quantize the RGB values into a regular $5 \times 5 \times 5$ bins to form a 125-d histogram for each superpixel. We then fit a multinomial distribution by normalizing the histogram after

weighting it with a Dirichlet prior, with hyperparameters set to 0.1.

The schedule also depends on assigning pixels to superpixels and therefore requires an initialization. We initialize our superpixel regions by a uniform grid, see Figure 5.2a. We set the *compactness* requirement such that $|\mathcal{K}| = 5$ and the minimum width of each superpixel to 1.

Finally, we can weight the energy terms for the boundary and region terms differently:

$$E(\mathbf{y}) = \sum_{p \in \mathcal{P}} \lambda_1 U_p(\mathbf{y}_p) + \sum_{p,q \in \mathcal{N}} \lambda_2 P_{pq}(\mathbf{y}_p, \mathbf{y}_q) + \sum_{p,q \in \mathcal{M}} Q_{pq}(\mathbf{x}_p, \mathbf{x}_q) \tag{5.4}$$

where the constants $\lambda_1 = 1$ and $\lambda_2 = 2$ were set by hand using a validation set. The effect on performance of different terms can be seen in Section 5.7.1. We set the pairwise terms based using the output of the boundary cost map, see Section 2.3.2.

5.6 Qualitative Evaluation

We first illustrate the iterative minimization of an energy using the *LatticeCut* algorithm. In Figure 5.2a-d we show four iterations of our algorithm. The evenly spaced grid gradually conforms to the contours of the images. Figure 5.11a shows the global cost function decreasing with each iteration of the algorithm. Figure 5.11b-e shows how the color model for superpixel 11 in Figure 5.2 evolves as the grid changes. For this Figure we used Gaussian mixtures for the colour models so that it is easier to visualize. At the first iteration, the superpixel contains both the skirt and water and there are two clear modes in the distribution. As the segmentation improves the superpixel conforms to the shape of the skirt and the colour model becomes unimodal and more concentrated.

LatticeCut converges with the same guarantees as *GrabCut* if we first minimize one orientation fully (so that each iteration minimizes the same energy globally) and then the second. However we minimize orientations alternately so that the shape of the superpixels is altered in both the horizontal and vertical directions over two iterations. This means that we do not fit the layers to the boundaries in the initialized superpixels (in this case uniform layers) and our final superpixels better fit the data. In practice we observe good convergence over several iterations, we use six iterations (three horizontal

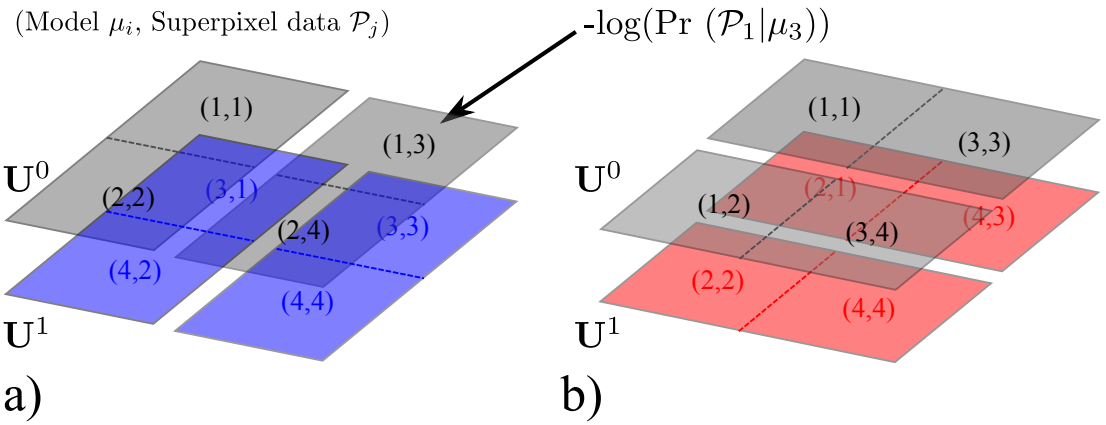


Figure 5.10: Model schedule. Layout of colour models in graph layers for 2×2 lattice. a) Costs by layer $\mathbf{U}$ for a binary vertical cut, given data $\mathcal{P}_j$ for superpixel j and model μ_i for superpixel i. b) Layout for horizontal cut.

and three vertical cuts). It would be possible to force convergence by checking the energy between iterations similar to *GrabCut* [225].

Figure 5.12-5.14 shows examples comparing the *LatticeCut* approach to our greedy approach from Chapter 3. In general we can see improved performance in regions of the image with weak boundaries but significant region color differences. Specific examples of where the method improves are discussed in Section 5.9.

5.7 Quantitative Evaluation

In this section we compare our *LatticeCut* (`LC`) algorithm to competing methods. Again we use the `MS` algorithm to merge superpixels from a higher resolution to a lower resolution to give us a benchmark for an algorithm without the topology constraint, referred to as `LC-MS`.

The images on the CamVid data are too large for us run the `LC` algorithm at full resolution. We therefore warp the images using the method presented in Section 4.3.3 and reduce the resolution by 50%. We then run the `LC` on this reduce resolution image before, re-scaling and re-warping. We refer to this algorithm as *adaptive lattice cut* (`ALC`) and the merged version as `ALC-MS`.

For the evaluation we use the same methodology presented in the Chapter 3. We use six performance measures, interpolate values for six superpixel resolutions where required, and summarize the results using the Borda score. Details can be reviewed in Section 3.5.2 and full results can be found in Appendix B.

Metric	F_{sd}					Cover Score				
Lattice Resolution / Algorithm/ Parameters	42	100	210	342	600	42	100	210	342	600
	6×7	10×10	14×15	18×19	24×25	6×7	10×10	14×15	18×19	24×25
LC (BEL) $\lambda_1 = 0\lambda_2 = 1$	0.321	0.356	0.438	0.532	0.547	0.673	0.732	0.788	0.814	0.834
LC (BEL) $\lambda_1 = 1\lambda_2 = 0$	0.351	0.411	0.535	0.610	0.680	0.683	0.741	0.791	0.823	0.842
LC (BEL) $\lambda_1 = 1\lambda_2 = 1$	**0.412**	**0.533**	**0.626**	**0.695**	**0.737**	**0.756**	**0.836**	**0.872**	**0.898**	**0.904**

Table 5.1: Potential terms. Experiment to show that both unary and pairwise terms improve segmentation performance. $\lambda_1 = 0\lambda_2 = 1$ - pairwise terms only. $\lambda_1 = 1\lambda_2 = 0$ - unary terms only. $\lambda_1 = 1\lambda_2 = 1$ - unary and pairwise terms. We can see that full energy function produces improved performance.

5.7.1 Potential terms

In this section we show that both the unary terms and pairwise terms contribute to the performance of the algorithm. We use a subset of 10 images from the BSD test set. Results for this experiment can be seen in Table 5.1 using the F_{sd} and CoverScore measures. We vary the parameters λ that weight the unary and pairwise terms and the results show that full energy function produces better performance across superpixel resolutions.

5.7.2 Comparing Boundary Maps

We investigate the three choices of boundary map using the BSD dataset [173]. Borda scores for competing methods can be seen in Figure 5.2. We can see a similar pattern to that presented in Chapter 3 where the `BEL` algorithm produces slightly improved performance. Plots of two measures can be seen in Figure 5.15.

5.7.3 Comparison to other algorithms

In this section we compare *LatticeCut* to other competing methods. Table 5.3 shows the Borda scores for competing methods. The `LC-MS` algorithm is ranked highest against competing methods on the BSD and VOC databases and the `ALC-MS` algorithm is ranked first on the CamVid dataset.

The improved performance of the `LC` algorithm over the `GRL` algorithm presented in Chapter 3 can be seen in Figure 5.16. We can see that `LC` dominates `GRL` at all resolutions. This improvement is despite the fact that the `GRL` algorithm can produce returning paths and demonstrates the benefit of a graph construction that exploits both

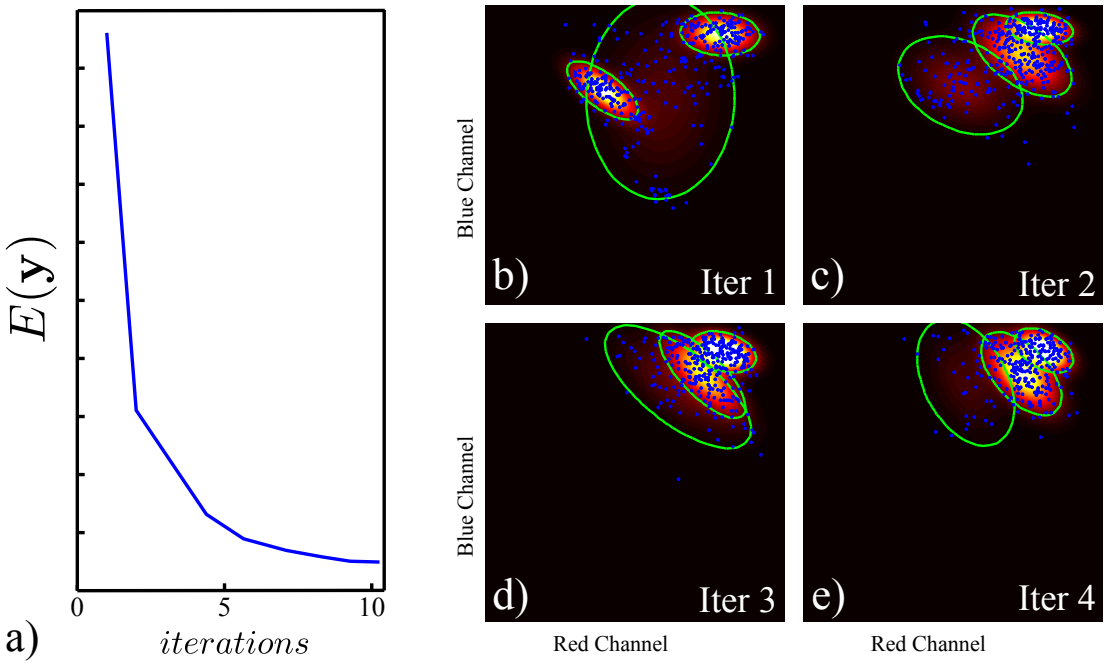


Figure 5.11: Iterative convergence of Lattice Cut. a) Convergence. The overall cost function decreases with every iteration. b-e) Local color models are computed by fitting a mixture of Gaussians to the RGB data within each superpixel. The RGB data and model from superpixel 11 from Figure 5.2a-d) for the first four iterations. As the segmentation improves, the color becomes more pure and the distribution more concentrated.

region and boundary terms.

Moreover, on the CamVid dataset the LC algorithm is also ranked higher than competing methods. This is an interesting result because the LC algorithm has the added topological constraint.

5.8 Runtime and Computational Complexity

The improved performance of the LC algorithm comes at the cost of increased runtime. Figure 5.17a shows the runtime for competing methods on the CamVid images against the number of superpixels including the time taken to run the boundary map algorithms. Our timing evaluations are based on an Intel(R) Xeon(R) CPU 2.49GHz with 4GB of RAM. Unlike the GRL algorithm presented in Chapter 3 a large part of our current implementation of the LC algorithm is in Matlab. This makes the timing comparison with algorithms coded reasonably efficiently in C++ difficult. However the main algorithmic component of our approach is the graph cut for which we use an effi-

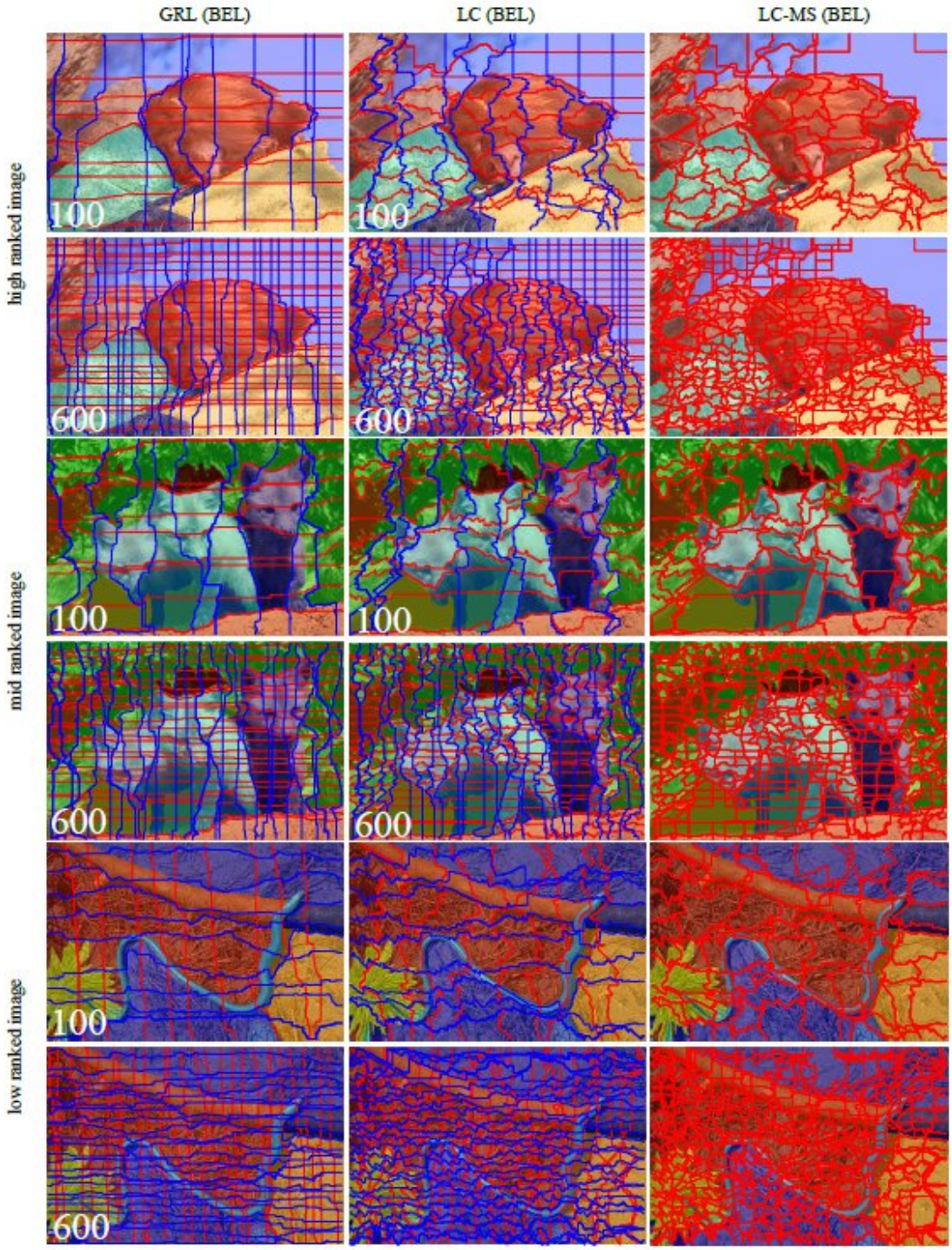


Figure 5.12: High/Mid/Low ranked images from BSD dataset comparing GRL and LC. Note the improved performance in regions with weak boundary information - eg. rock to bottom right of lion in the top two rows. Specific failure modes are shown in more detail in Figure 5.20. Images can be comparred to Figures 3.21 and 3.22.

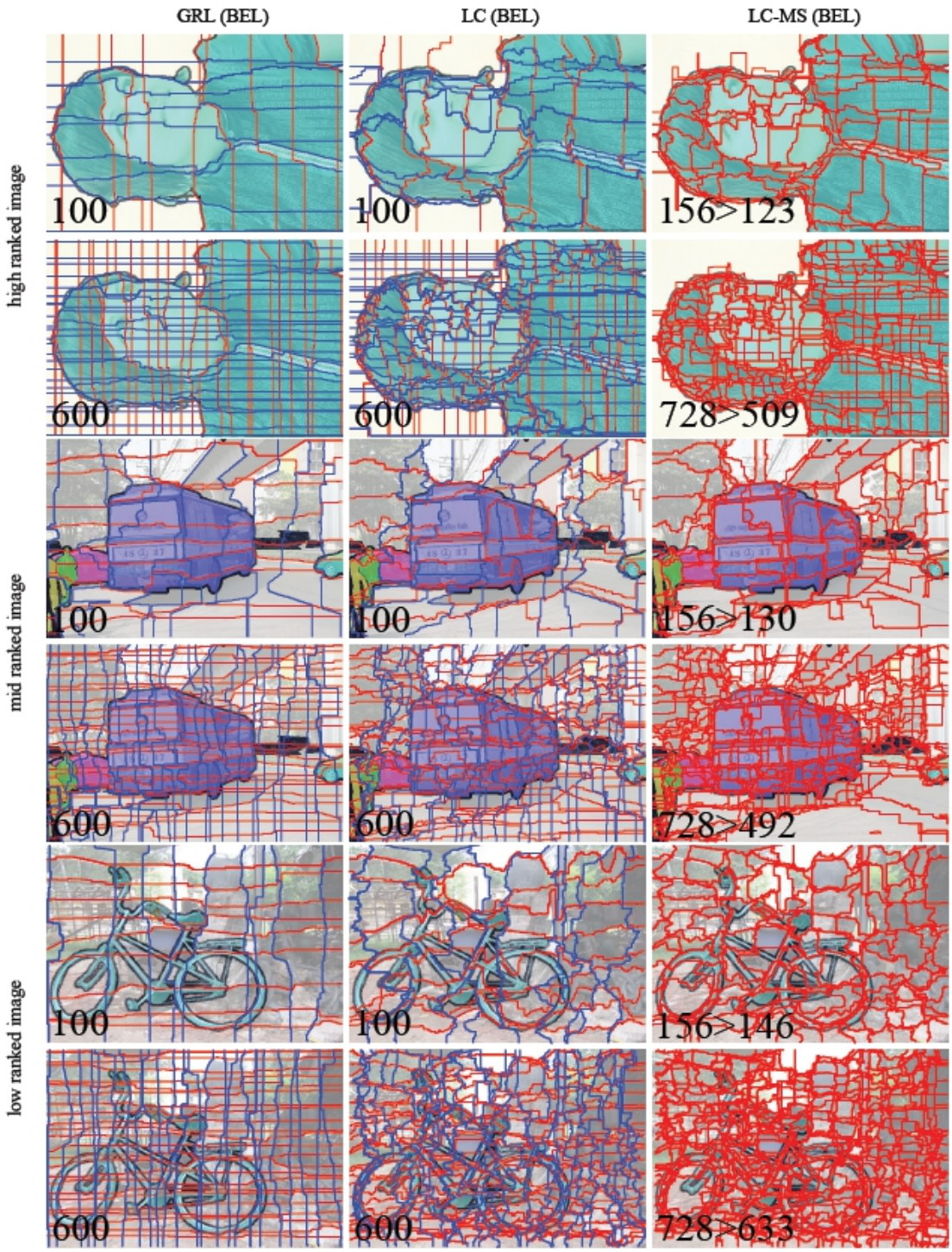


Figure 5.13: High/Mid/Low ranked images from VOC dataset comparing GRL and LC. In general we see better alignment to object boundaries. For example, on the curvature of the bicycle wheel or the shadow of the coach. Images can be comparred to Figures 3.23 and 3.24.

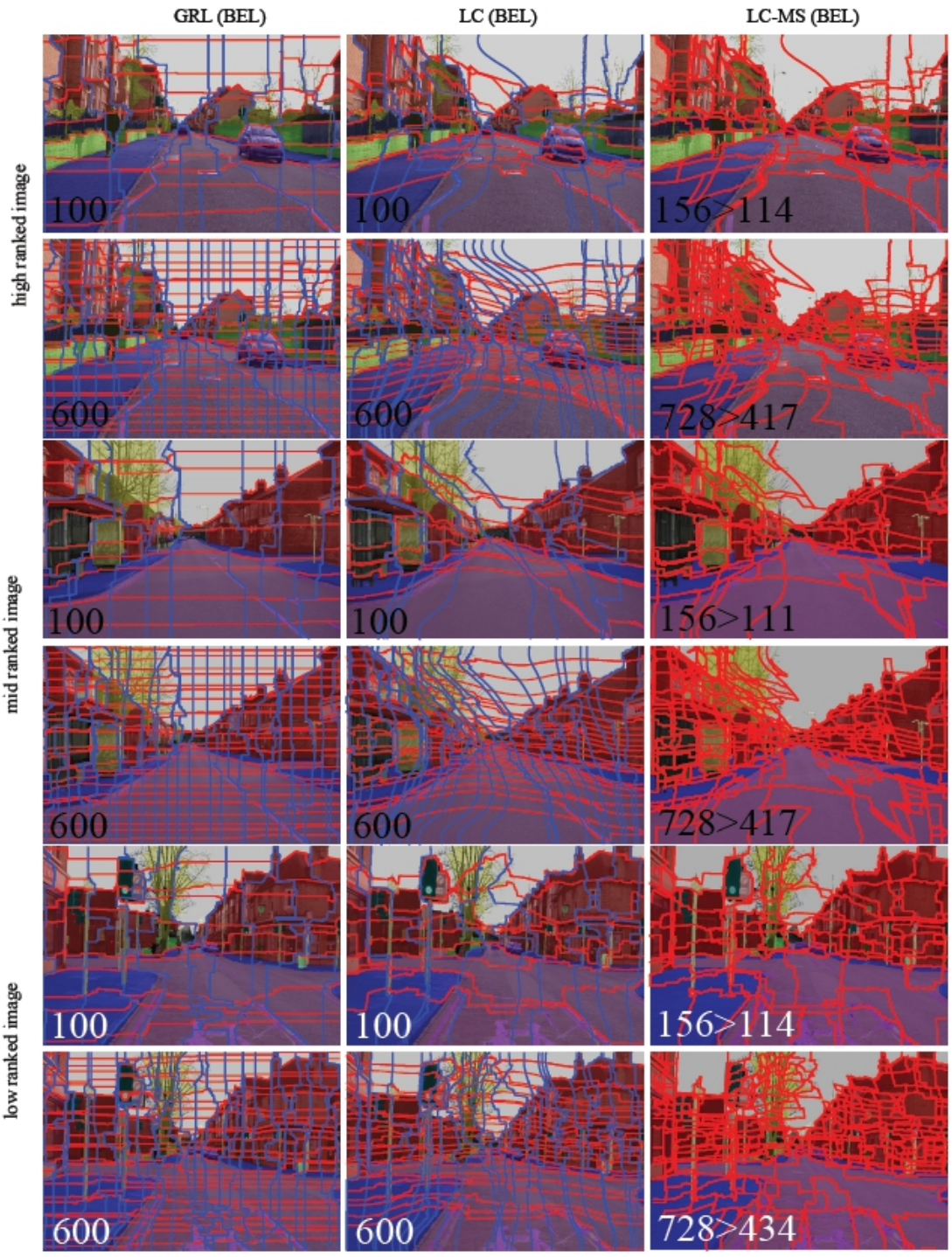
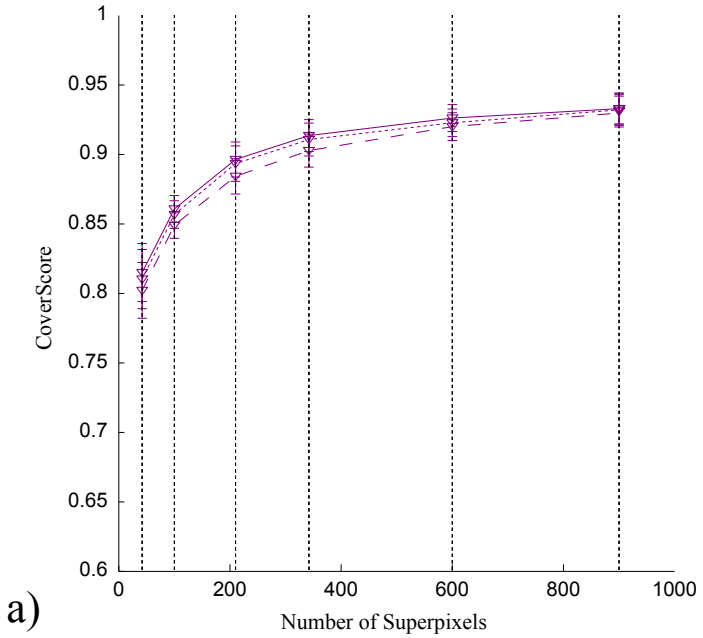
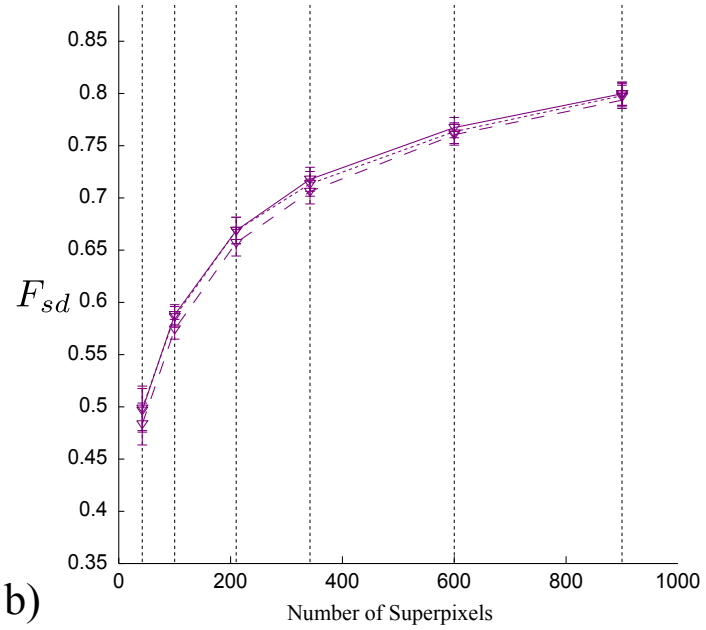


Figure 5.14: High/Mid/Low ranked images from CamVid dataset comparing GRL and LC. In general we see better alignment to object boundaries. For example, on the separation of pavement an wall on the left hand side of the high ranked image. Images can be comparred to Figures 3.25 and 3.26



a)



b)

KEY: — BEL [74], - - - Pb [174], – – Canny [43]

Figure 5.15: Comparing performance of LC using different boundary maps on BSD. a) Number of superpixel vs Cover Score. b) Number of superpixel vs F_{sd}. BEL algorithm produces slightly improved performance.

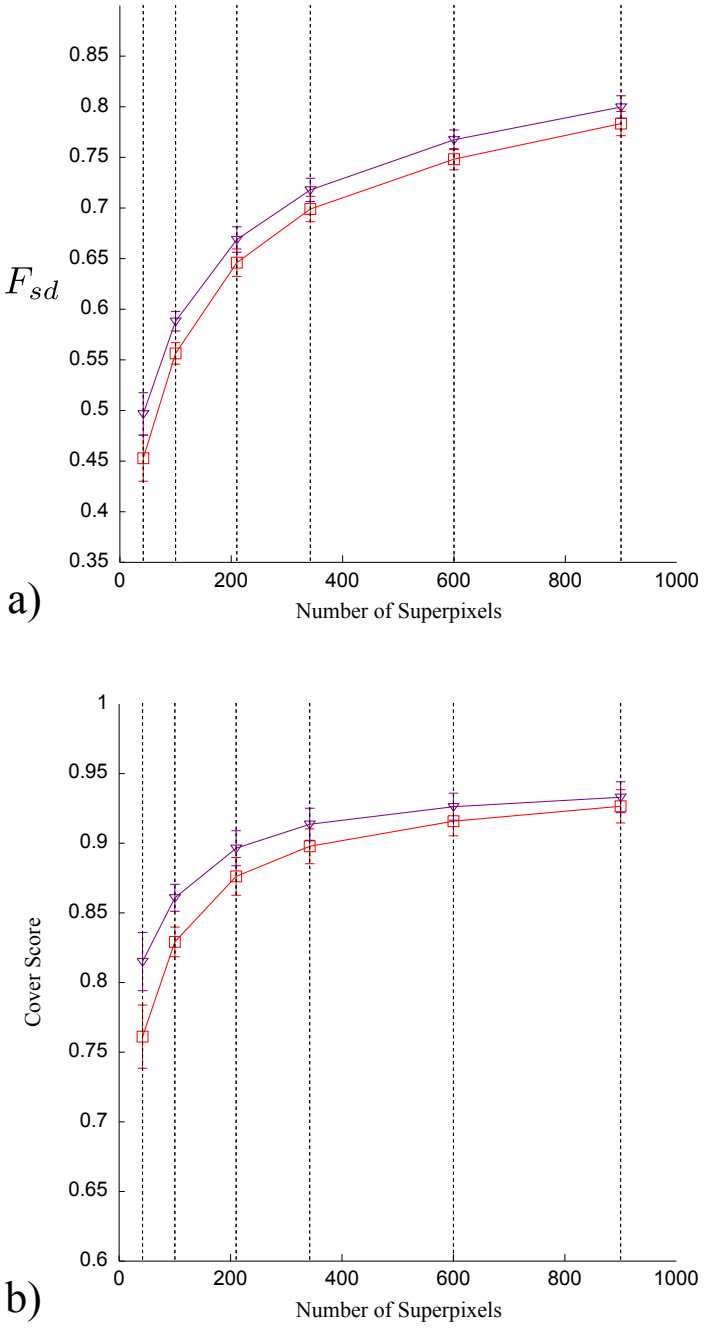
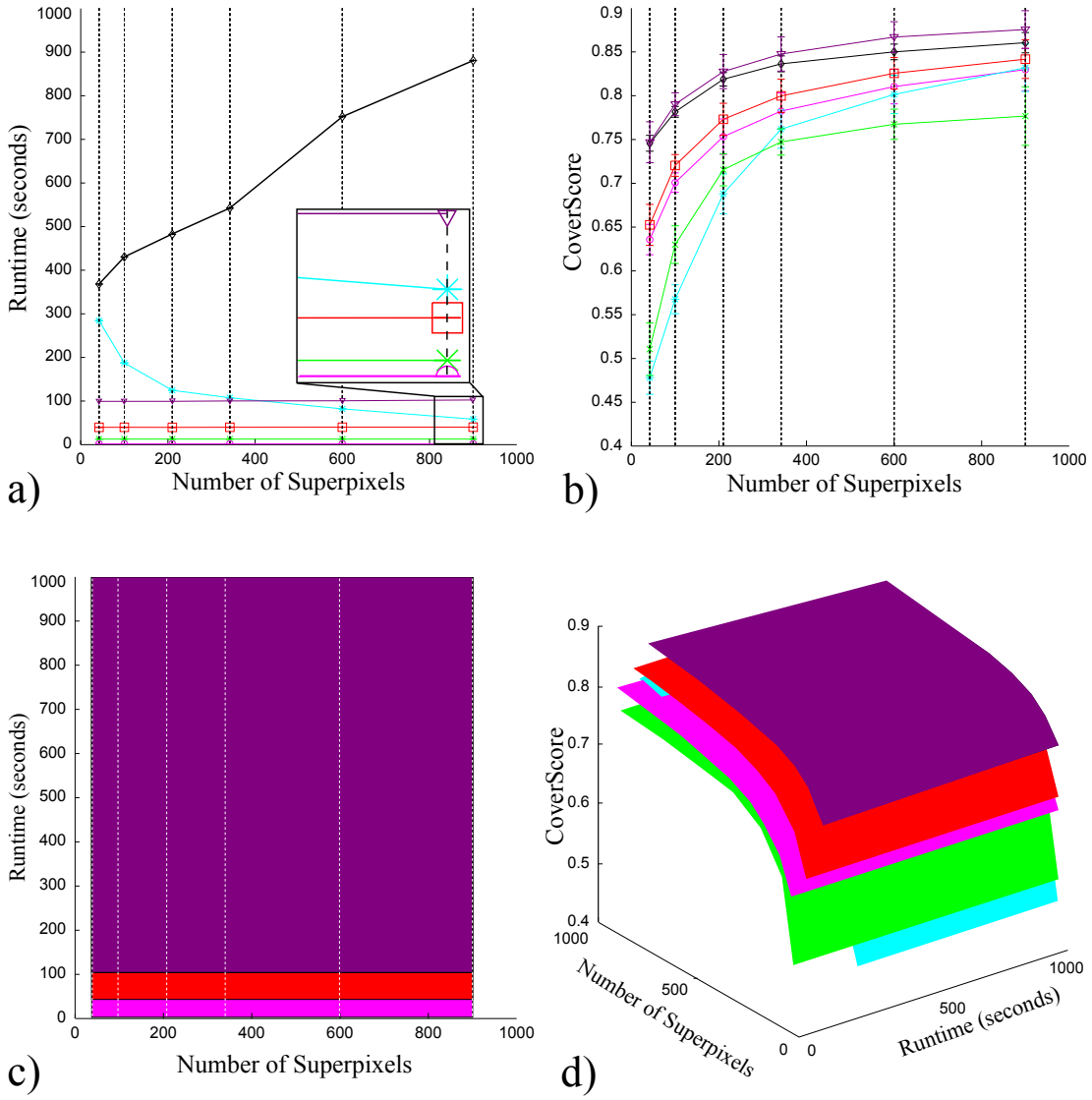


Figure 5.16: Comparing performance of GRL presented in Chapter 3 with Lattice Cut (LC) on BSD. a) Number of superpixels vs F_{sd}. b) Number of superpixel vs Global Consistency Error. LC improves performance at all superpixel resolutions.



KEY: Uniform, MS [53], FH [88] NC [185], GRL(BEL) [182] LC(BEL)[184]

Figure 5.17: Comparison of algorithm runtime. a) Runtimes for competing algorithms on the CamVid dataset including both the time taken to compute the boundary map and the time for segmentation. b) Cover score vs number of superpixels on CamVid dataset. c) A 2D projection of $CoverScore \times Numberofsuperpixel \times Runtime$ space with the Cover Score axis going into the page. Note our improved performance on the hull using the LC algorithm. However, GRL still dominates at lower runtimes. d) Full 3D projection of $CoverScore \times Numberofsuperpixel \times Runtime$ space.

Algorithm	Reference	Dataset
		BSD [173]
LC (BEL)	[184] ([74])	**108**
LC (Can)	[184] ([43])	106
LC (Pb)	[184] ([174])	99

Table 5.2: Boundary map ranking using GRL algorithm with different boundary detection algorithms. Borda scores shown for three datasets (columns 1-3) summed across both metrics and resolutions. A total score across datasets is given in the last column. BEL algorithm is ranked first on all datasets. Maximum Borda score on individual dataset = 108.

cient implementation [36]. The time we use for comparison is therefore the time taken for the graph cut, averaged over the six iterations, with an additional 10% added to estimate an efficient implementation of the LC bookkeeping. Therefore the curve in Figure 5.17a gives an indication of runtime performance with the number of superpixels but the absolute value should only be taken as an estimate.

Similarly to Chapter 3 we can again consider the hull in the fitness/cost space where we consider runtime as a cost function. A 2D projection of the $CoverScore \times Numberofsuperpixel \times Runtime$ space (with the Cover Score axis going into the page) can be seen in Figure 5.17c. Here we can see the performance trade off between LC and the GRL algorithm we introduced in Chapter 3. The additional performance comes at the expense of slower runtime.

The complexity of the LC algorithm, it is dominated by the st-MINCUT algorithm. We use the st-MINCUT implementation made publicly available by Vladimir Kolmogorov based on published work [36]. The worst case run time for this algorithm is $\mathcal{O}(VE^2|C|)$ where V is the number of nodes, E is the number of edges in the graph and $|C|$ is the cost of the minimum cut in the graph. This run time complexity is worse than other standard approaches [73] but has been shown to empirically outperform perform other approaches on typical problem instances in vision [36].

For an image with $\sqrt{N} \times \sqrt{N}$ pixels and $|\mathcal{L}| = S + 1$ labels each application of *LayerCut* uses a graph with approximately NS nodes. This gives a runtime of $\mathcal{O}((NS)^3|C|)$ where we have assumed that $E = \mathcal{O}(V)$. However our approach uses

Algorithm	Reference	Datasets			
		BSD [173]	**VOC** [87]	**CamVid** [39]	Total
FH	[88]	118	151	125	394
GRL (BEL)	[182] ([74])	175	204	188	567
GRL-[MS] (BEL)	[182] ([74])	187	223	212	622
ALC (BEL)	[184] ([74])	-	-	245	-
ALC-MS (BEL)	[184] ([74])	-	-	**247**	-
LC (BEL)	[184] ([74])	235	232	-	712
LC-[MS] (BEL)	[184] ([74])	**245**	**238**	-	**730**
MS	[53]	192	205	136	533
NC (Pb)	[186] ([174])	236	237	233	706

Table 5.3: Algorithm ranking using Borda score on three datasets and combined totals. The LC algorithm performs well and the merged version is ranked first on all datasets. ALC uses the whole-image warp presented in Chapter 4. Maximum Borda score on individual dataset = 252.

a subset of labels $\mathcal{K} \subset \mathcal{L}$ at each pixel so in practice S remains constant as the number of superpixels increases. Additionally, a runtime of $\mathcal{O}(N^3|C|)$ for maxflow algorithms is seldom observed in practice. This runtime may be compared to a runtime of $\mathcal{O}(S\sqrt{N}\ln(S\sqrt{N}) + 4S\sqrt{N})$ for our greedy approach in in Chapter 3. Section 5.10.5 discusses several techniques to improve the runtime of the st-MINCUT algorithm.

5.9 Failure Modes

Figure 5.2 illustrates one major limitation of our method: **Constraint 2** means that the vertical component labels always increase as we move across the image and likewise the horizontal component labels always increase as we move down the image. This restricts the possible shape of the resulting superpixels as their boundaries can never turn back on themselves. An example of where this reduces performance can be seen in superpixel 7 of Figure 5.2d. This superpixel cannot expand upwards under the arm region or the black boundary between pixels $\{6, 7\}$ and $\{10, 11\}$ will turn back on itself. This would create a scanline with the illegal label ordering 0-1-2-1-2-3.

However, the LatticeCut algorithm does improve performance over the GRL algorithm. Selected examples of the improvement may be seen by comparing the results in Figure 3.40a-d.

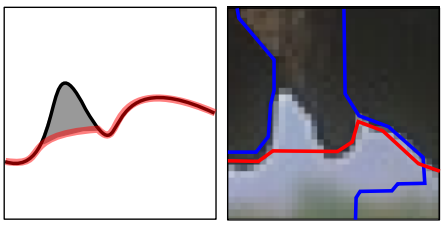


Figure 5.18: Shrinkage error.

We can see that the color models used in the unary terms tend to produce better estimates of regions and therefore "close off error" like the bo-tie effect are reduced. Additionally, there are no artifacts due to the greedy method that uses image strips, see Figure 5.20d.

However, the remaining six errors introduced in Section 3.9 are not improved significantly. These include common 'shrinkage bias' effects [37, 140], see Figure 5.18. There are also errors associated with the separation of non-distinct or porous classes, see Figure 5.19. A more detailed discussion of these effects along with alternative examples can be found in Section 3.9.

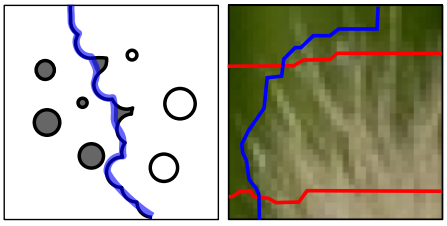


Figure 5.19: Porous error.

5.10 Discussion and Future Work

The graph construction that we introduce in this chapter makes use of several ideas from the literature. We draw comparisons between our method and other techniques before setting out several possible extensions and improvements to our current method.

5.10.1 Comparison to existing methods

We set up the *LayerCut* problem as a multi-label MRF on a graph in the same manner as the construction of Ishikawa [128]. However, Ishikawa only uses constraint edges in the columns along side data terms. This has the effect of constraining the cut on the graph to correspond to a valid configuration but the only restrictions on the configuration are as a result of convex priors. This approach to minimizing a multilabel MRF is extended for general convex priors by the work of Schlesinger and Flach [231].

Li et al. [161] show how to construct a set of optimal layers using a minimum path algorithm. They formulate their problem as a closure set problem on a graph with a set of nodes for each label. This is shown to be equivalent to using the st-MINCUT algorithm to find a set of optimal nested surfaces in a different graph by Delong and

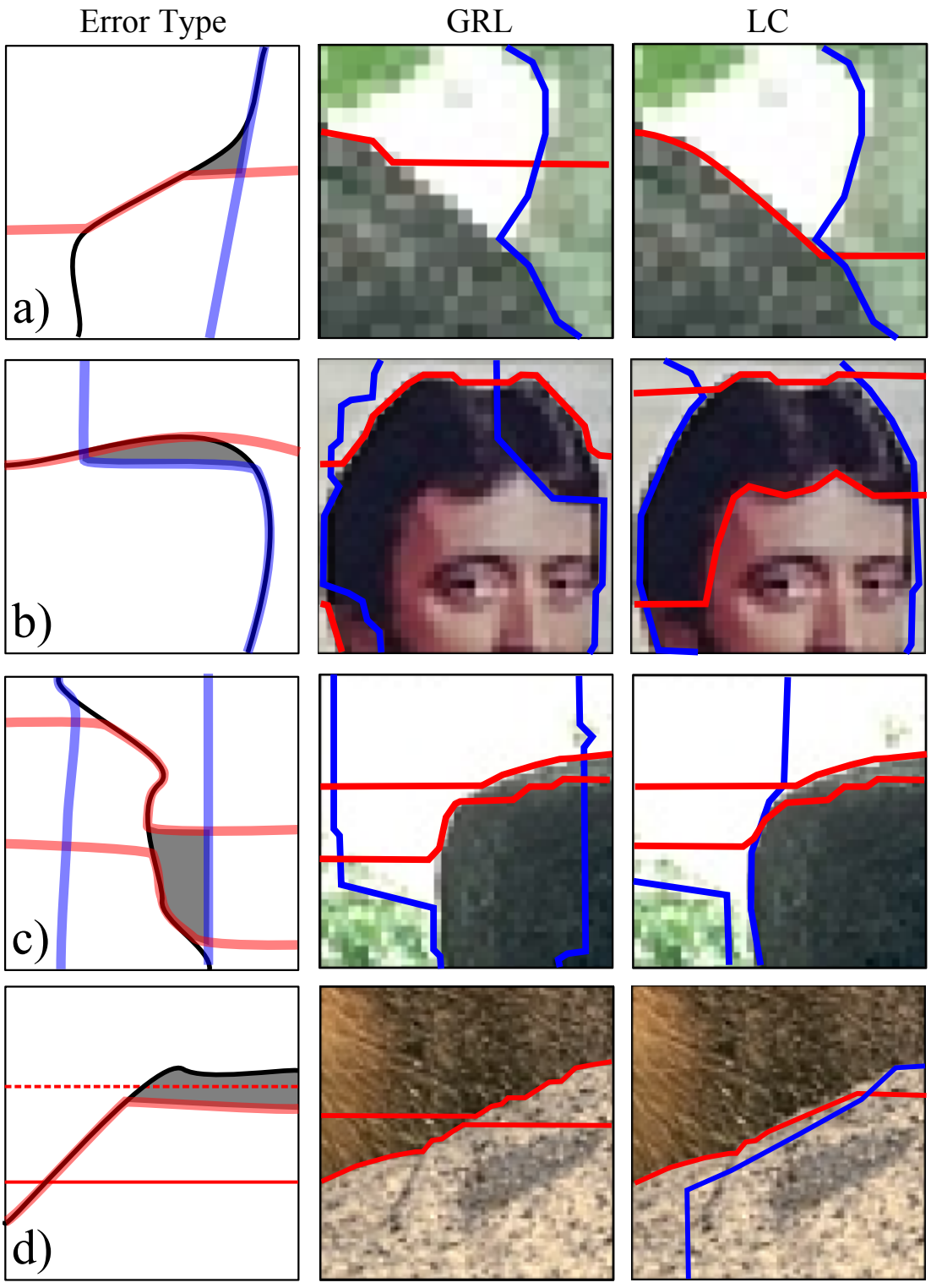


Figure 5.20: Failure Modes. First column: A schematic for error types with object boundaries in black and poorly segmented regions coloured gray. Second column: GRL algorithm with horizontal and vertical boundaries in red and blue respectively using real examples from the BSD dataset. Third column: LC algorithm. The LatticeCut formulation reduces artifacts based on failure cases a-d.

Boykov [67]. However, the graph of Li et al. [161] contains no unary terms whereas our construction includes both unary and pairwise terms with the additional restriction on the shape of the boundaries between layers.

In a series of papers Liu et al. [166, 167] present work on order preserving moves, which originally inspired the work presented in this chapter. They use constraint edges in the pairwise terms to enforce the ordering of labels in either the vertical or horizontal directions in an expansion framework that chooses between several labels at a time - rather than the usual binary expansion. Liu et al. [166] resort to an expansion move because the labels in their geometric class problem cannot be linearly ordered and therefore the terms are not *submodular*. Our approach differs in several respects. First, we divide our labelling problem into two sets of interacting labels for both vertical and horizontal layers. In each sub-problem the labels are linearly ordered which means we can find the global minimum with one cut. Furthermore, we include constraint edges between layers in our construction to enforce a sequential ordering. Lastly, **Constraint 4** in our approach allows the solution of one set of labels to constrain the solution to the second set of labels, which is a technique absent from their approach.

Motivated by our earlier publication on superpixel lattices [182] Sargin et al. [229] introduce a factor graph that uses a layer representation to optimize a global cost function. Their method uses the Turbo decoding algorithm of Perronnin et al. [206] for inference. Whilst this method minimizes a global cost function (unlike our original divide and conquer approach) and is very efficient it does not have any convergence guarantees. It would be interesting to compare their approximate optimization method directly to the method presented in this chapter but would require significant additional work.

Lastly, subsequent to the publication of this work [184] there has been ongoing progress in applying different MRF representations to the problem of superpixels and supervoxels [268]. This work uses an iterative estimation of the label field using α-expansion but does not set out to impose the constraints of the regular lattice and therefore has no limitations on the number and topology of the labels. Benchmarking this algorithm against competing methods should be included in future work.

5.10.2 Initialization

One aspect of our approach that requires significant further investigation is the effect of initialization on the final results. The method presented in [268] also relies on an initialization scheme for assigning potentials to superpixels. All results presented in this chapter use a uniform grid for initialization but the algorithm could equally well be initialized randomly because the constraints force a final lattice on the label field. Investigating the stability and convergence properties of the lattice under different initializations would be useful. Moreover, it would be possible to make a direct comparison of performance with ([184]) and without ([268]) the constraints using identical initializations. This is left for future work.

Importantly, the work on *Scene Shape priors* in Chapter 4 indicates one way in which it may be possible to initialize superpixel algorithms with a set of (non random) superpixels to produce more favourable results - ie. by changing the initial distribution of size and shape of superpixels before running the *LatticeCut* algorithm. The implementation of the adaptive lattice in this chapter used the warp method presented in Section 4.3.3 to alter the image before sub-sampling. However an alternative way in which this constraint could be imposed would be to alter the *compactness* criteria such that the cross section shown in Figure 5.7b was non-linear. This would encourage a greater number of superpixels in one region of the image over another and produce a foveation in the final segmentation.

There is also the question of how to initialize a lattice of superpixels using ground truth data. For instance, if you wanted to learn prior distributions over a lattice of superpixels in a similar manner to algorithms reviewed in Chapter 2 you might want to start from ground truth lattices. Similar to the experiment using ground truth boundaries (Section 3.8) you would like to be able utilize both ground truth boundaries and models trained solely on data from within ground truth regions. However, you would not want to incur the cost of human labelling both ground truth for object classes and the lattice. Furthermore, it is unclear whether there would be good agreement between human subjects on how this was to be done. One approach to this problem would be to use seeds based on supervised segmentation [225]. Further work would involve improving the lattice properties based on some simple toy examples to highlight different failure modes in a supervised manner.

Lastly, an interesting thing to note about convergence of the *LatticeCut* algorithm is that **Constraint 4** need not be maintained at every iteration. For instance we could separately solve for both the vertical and horizontal layers and allow superpixels boundaries to act independently from each other. This would likely mean that superpixels in different orientations both fitted similar object boundaries or results like that illustrated in Figure 5.9b are found. However, producing superpixels that have a good alignment with image data before imposing the constraint may be useful in guiding the design of heuristics to deal with difficult cases. For example, you might be able to test for foreground/background and decide which orientation to prefer locally before imposing the constraint and iterating to a final solution.

5.10.3 Graph construction and higher order terms

It is possible to augment our approach using several other techniques developed in the literature. The work of [67] introduces some interesting possibilities. The authors use high cost edges, that they call *interaction* terms that play a similar role to *constraint* edges, where instead of restricting the set of feasible solutions they simply make them unlikely. For instance, they use an *attraction* term to penalize the size of a region. We might use a technique like this to overcome some of the limitations of the non-returning boundaries that our approach introduces. For example, we could have a *LayerCut* algorithm that has a fixed set of layers with additional partial-layers that could appear by paying a high fixed cost. This would produce a result similar to Figure 5.8e where the grey label can appear between the black and white labels. These partial layer models could either be kept separate or be made redundant so that the partial layer takes on the model of the superpixel to its left or right. Partial layers may be a useful way of dealing with articulation or occlusion effects that do not inherently fit the current non-returning boundary restriction (See Figure 5.8e). Other possibilities include estimating the numbers of layers (and therefore superpixels) in either an *a priori* manner [65] or based on the estimation of objects and scene geometry [120].

There is a large body of work on high-order MRFs (those with a clique size greater than two) both theory [224, 152, 136, 228] and application [148, 139, 248, 284, 150]. Ramalingam et al. [215] extend the result presented in [141] and give the graph construction for characterizing a general k^{th} order multi-label energy function in a similar

graph to the one we have used in this chapter. It may therefore be possible to include high-order terms that characterize properties of the superpixel as a whole. For instance, one high-order cue might be an estimate of texture based on a learned dictionary of patches [136].

The *LayerCut* formulation can be also be readily extended to multiple images. This could mean hierarchical, temporal or stereo models for multiple images. The temporal model would be similar to that introduced in Section 3.10.

5.10.4 Multiple Segmentations

Producing superpixels based on inference in the MRF framework offers the possibility of re-visiting work on multiple segmentations. Currently, generating multiple segmentations has been largely based on random or heuristic sampling from the parameter space of superpixel algorithms [227, 121, 172, 139]. However, there are methods that address measuring uncertainty within st-MINCUT solutions [135] which may serve as a basis for approaching multiple segmentations in a principled manner by computing the M most probable configurations of the MRF [288]. Sampling from directed versions of the model [75] could also be one possibility for exploring multiple segmentations.

5.10.5 Efficiency

Work on measuring the uncertainty in an MRF labelling [135] requires methods for efficiently recalculating the st-MINCUT on updated versions of the graph. Additionally, the suggestion of moving to graphs with a greater number of nodes or those with high order cliques also increase the memory and runtime requirements. Future work should investigate exploiting new methods for improving the computational and memory efficiency of st-MINCUT solution [66, 16, 17, 165, 247].

5.10.6 Potential functions

The framework presented in this chapter is independent of the representation of colour, or texture, used to characterize image regions. Future work should explore the use of more complicated potential functions that include texture, shape, and class information for performing inference on the compact region of support that superpixels provide [239, 248, 209, 150, 276].

For example one possible line of enquiry would be to revisit work on layout con-

sistent random fields [283, 123] using potentials based on weak estimates of object parts. These might be well represented using graphs that preserve a set of ordered layers.

One important factor that was not exploited in the work in this chapter was that of learning the set of parameters on the potentials as they were set by hand. Several authors pursue a piecewise learning scheme for finding appropriate weights in a CRF formulation [249, 239]. Other approaches include *contrastive divergence* [117, 115, 256], work on simple approximations of the partition function [146], or an iterative graph cut scheme to learn the weights in a CRF based on a training dataset [250].

Lastly, the assignment of unary potentials to superpixels is a model selection problem. In this chapter we have used one model for each superpixel but a learning approach could be used to estimate the number of models required - ie. which superpixels share models. Future work based on estimating the number of models, or clusters [211], would give a more principled approach to merging superpixels than the current application of `MS` used in the pervious chapters.

5.11 Conclusions

We have shown how to impose an ordered set of layers on the label field of a multi-label MRF and used this to produce a superpixel lattice algorithm that is optimal in either the vertical or horizontal direction. Moreover, we have shown that this new approach outperforms our greedy implementation presented in Chapter 3 and on some datasets outperforms competing methods despite the added topological constraint. This improvement may be attributed to using a formulation with both unary and pairwise terms and is achieved despite using only simple color models for the unary terms.

This improvement in performance comes at the expense of greater complexity and runtime. However the ubiquity of graph cut problems in vision suggests there will be ongoing improvements in efficiency [66, 16, 17, 165, 247]. More importantly, by tightly integrating the estimation of a compact region of support with a method of inference the superpixel representation itself need no longer assume the role of a simple pre-processing step.

For example, our method has several advantages over existing superpixel approaches: The representation of the image as a distributed collection of models with

known relationships suggest that it may be possible to incorporate ideas from *pictorial structures* [89], *constellation models* [94, 278, 42] and *layout consistency* [283, 123]. Recent work that takes this approach to segmentation, labelling ordered sets of regions, includes [92].

Moreover, our approach is naturally iterative and may have advantages over fixed quantizations of the image space. For example, Hoiem et al. [120] demonstrate improved estimation of occlusion and surface features based on iterative estimation of the region of support in a CRF framework.

Chapter 6

Conclusion

In this chapter we summarize the findings of previous chapters, highlight outstanding problems and discuss the direction of future work.

6.1 Summary of Findings

In Chapter 3 we introduced the notion of a superpixel lattice and presented two greedy algorithms that produce a solution. We demonstrated good segmentation performance on standard datasets. The evaluation incorporated several measures of performance, including one that we introduced, and we showed that if runtime is considered our algorithm dominates large areas of the performance hull.

We demonstrated improved performance of our algorithm using modern learned edge detectors over the traditional Canny edge detector. We also made use of a standard implementation of the Mean Shift algorithm [53, 176], enabled by the lattice structure of our segmentation, to improve performance by merging superpixels. Finally, we showed how to adapt the algorithm to improve temporal stability.

In Chapter 4 we observed that the different distribution of objects within different datasets produced varying distributions of object boundaries. We showed that it is possible to learn low resolution priors on the distribution of object boundaries given labeled training data and we demonstrated how to use this to improve segmentation performance on a particular dataset [39].

Finally, in Chapter 5 we recast the superpixel lattice problem as a coupled Markov Random Field. We demonstrated improved performance over our earlier greedy methods, at the cost of greater algorithmic complexity, and discussed how the framework may be extended in several directions. One quantitative demonstration of the progress

we have made during the course of the thesis can be seen in Figure 6.1. In Figure 6.1a we show the performance improvement on the CamVid dataset with algorithms from successive chapters. Figure 6.1b shows an example of the performance for the merged lattice from Chapter 5 against the best two competing methods.

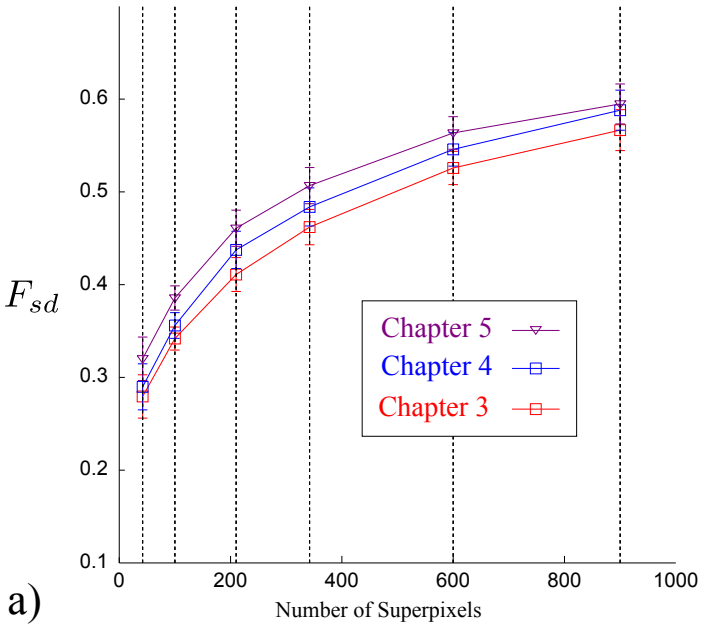
6.2 Limitations and Future Work

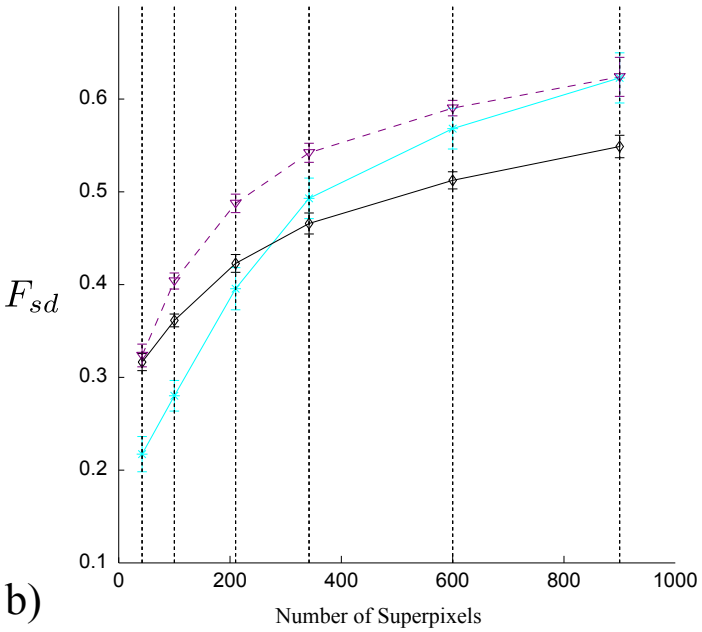
Research into superpixels is ongoing and new algorithms are regularly introduced. Recent superpixel algorithms include [159, 55, 11, 268]. However none of these methods have the fixed topology of the methods presented in this thesis.

Our evaluation includes a representative sample of competing algorithms. It should be noted that the parameters used for competing methods and the algorithms we present in Chapters 4 and 5 were set by hand or searched heuristically. There are several techniques that may be employed to try and improve results for all methods based on search of the parameter space [86].

Our experimentation demonstrates that the lattice can achieve comparable or better performance than competing methods - although we must relax the lattice constraint to achieve best performance. Moreover, the algorithm presented in Chapter 3 has already been successfully included in the vision pipelines of other works in the literature. Warrell et al. [275] use a superpixel lattice as part of their epitome prior model for labelling problems to make inference efficient. Choi et al. [51] base feature vectors for a CRF classifier on a region of support using regular 8×8 neighbourhoods on a superpixel lattice. Recently, Mu et al. [189] use a superpixel lattice as the basis for their semantic object cutout algorithm for an image editing system. Their implementation is used to speed up interactive segmentation in a similar manner to LazySnapping [163]. These applications exploit the efficiency and ease of engineering that our approach delivers. Furthermore, the work in Chapter 5 offers the prospect of new directions for research that include ordering constraints at the level of inference. Several notable directions this research might take are discussed at the end of Chapter 5.

However, there are other fundamental questions that need to be addressed. The notion of a *superpixel* itself needs clarification. There are several studies that suggest that the selection of features of *intermediate complexity* are optimal for classification tasks [263, 114] and the size of useful pixel clusters is likely to be application depen-





KEY: Greedy Regular Lattice [182] Adaptive Regular Lattice [183] Adaptive LatticeCut (ALC) [184] ALC-[MS](BEL) [184] Mean Shift (MS) [53], Normalized Cut (NC) [185],

Figure 6.1: Lattice Improvement. a) Improvement of lattice algorithms during chapters of the thesis using the CamVid dataset [39]. b) Example of merged Adaptive LatticeCut on CamVid data against competing methods.

dent. Perhaps a hierarchy of pixel cluster sizes will be required to capture different classes and textures. One naming convention would be to use the ratio of the clusters to either the object or image size, leading to an SI unit scale of *hectopixels*, *kilopixels*, *megapixels* and so forth.

Considering this scale leads naturally to the problem of partitioning objects into parts. At the lowest level, textures seldom conform to a simple square lattice which would suggest that at this level of the hierarchy the modeling power of a regular grid is reduced - though not necessarily the engineering convenience. Indeed, the lattice representation seems to gain most of its power by being conservative i.e.not always producing very accurate object boundaries - but always capturing a useful sized region of the object, in a uniform way across the image. As we move higher up the scale it is also hard to see how a lattice could produce satisfactory results at the pixel level with complex structures like bicycle spokes or porous classes such as foliage that simply do not conform to a lattice structure. Therefore it is perhaps best understood as useful intermediate representation of the image pixels that can facilitate fast and tractable inference [206, 91, 47, 229, 44, 233].

Additionally, formulation of the algorithms on a graph allows the framework presented in the thesis to be applied to surface meshes or space-variant images [271, 110] which may be useful for alternative applications. It also suggests exploring different lattice symmetries eg. triangular lattices [64], and more generally to ask when lattice parameterizations of objects and scenes are most useful? It is unclear that as the number of classes and scene complexity increases that having an approach with specific articulated part-based models [94, 143, 89] will scale appropriately and a more general approach based on lattice parameterizations may prove useful.

6.3 Final Remarks

Computer vision has enjoyed many recent successes and is now routinely encountered as a technology from film production [1] to consumer products [226]. However, it is common for the state spaces of vision problems to be very large and a recurring theme is how to compute practicable solutions using efficient algorithms [93].

Object segmentation remains one of the most challenging and fundamental problems in computer vision. Global optimization of 2D labellings are usually NP-hard

[38] and the importance of identifying solutions that are polynomial should not be underestimated. Even widely used move making algorithms [38, 167, 137, 156, 155] that can obtain known bounds on strong local minima can produce poor results on certain problems [122, 67, 167].

There remains a large amount of theoretical work to be done on exploring and categorizing 2D label fields with restricted topology that can be solved exactly in polynomial time. The contribution in Chapter 5 produces a set of ordered layers with non-returning boundaries. Other recent work includes nested containment [67] and tiered label sets [92]. Hopefully this direction of research will lead to new tractable, robust models for objects and scenes.

Appendix A

Path constraints

We saw in the middle of Chapter 3 that to force a lattice on the segmentation using greedy shortest paths we need to alter the graph at each iteration. The result we would like to achieve for parallel paths (paths in strips of the same orientation) can be seen in Figure A.1a. To do this we must restrict the set of feasible solutions to the shortest-path problem by altering the cost of *constraint* edges in the graph. The notation used in this chapter to describe shortest paths is adopted from the standard treatment presented in Cormen et al. [54], Chapter 24, pages 581-619.

These constraint edges limit the search space of each shortest-path to nodes that are not included in previous paths. The example presented in the earlier chapter can be seen again in Figure A.1b. Note that this constraint does not force the order of the final paths. It would be possible that two parallel paths could be inserted in the overlap region of two adjoining strips in the opposite order to the strip themselves. An example of this is illustrated in Figure A.1c, however this does not effect the final structure of the lattice which is governed by the number of paths, not their order. We use the term *legal path* to describe one that obeys the correct connectivity of the graph - not necessarily the constraint edges.

Secondly, we would like orthogonal paths to cross cleanly once. An example of the result we would like to achieve can be seen in Figure A.1d. We must prevent subsequent paths following existing orthogonal paths, see Figure A.1e, so that a single superpixel cannot be split into parts that are separated. Additionally, crossing once ensures there are not small bits of superpixels that need to be merged arbitrarily in order to preserve the total number of superpixels, see Figure A.1f.

To achieve these goals we alter the weight of edges in the graph after the addition

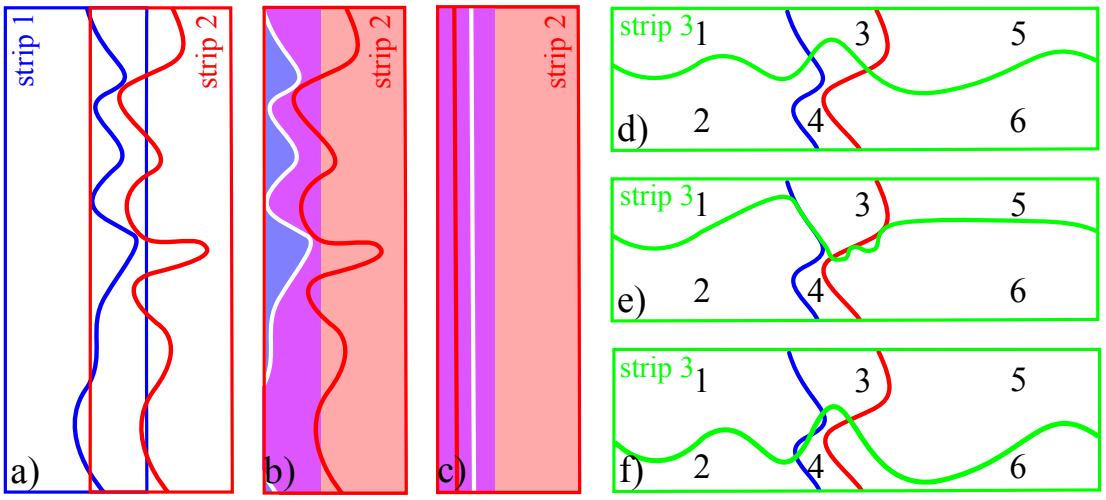


Figure A.1: Example Path Crossings. a) Two parallel vertical paths in adjacent overlapping strips. The overlap means that the search space of the shortest paths overlap considerably. b) To prevent the path of the second strip (red) crossing the shortest path found in the blue strip we must remove edges from the white region. The region of nodes that appear in both searches, the strip overlap, are coloured purple. Nodes that are now un-reachable for a path in strip 2 are coloured blue. c) Example where two parallel paths occur in reverse order to the strips, ie. the path in strip 2 occurs to the left of the path (white) in strip 1. d) Orthogonal paths must cross only once. e) We must also prevent paths in orthogonal strips following the same boundaries. f) Lastly, we must ensure that no additional superpixels are created by paths crossing multiple times.

of each path. The edges that we alter restrict the set of feasible solutions for subsequent shortest paths on the graph. We refer to these edges as *constraint edges* [128]. Let us begin by setting a large constant as the sum of all edge weights in the original graph, $\gamma = 1 + \sum_{(u,v)\in E} w_{uv}$. This constant is greater than any existing path in the graph. In general we would like to have to pay this large fixed cost every time we cross an existing path.

We can make this more formal by separating the nodes from each strip into three sets. Set one, labeled 1, is on one side of an existing shortest-path. Set two, labeled 2, is on the opposite side of this path. The third set is the set of nodes $p = \langle \nu_0, \nu_1, \ldots, \nu_k \rangle$ that belong to the existing path that imposes the constraints.

To discuss the cost of possible shortest paths we introduce some notion. Let $\rightsquigarrow$ indicate the *reachable* relation between two sets, let $\rightarrow$ be adjacency between nodes

and $\nrightarrow$ be non-adjacency. A path from a node in set 1 to another node in set 2 with a weight of δ is denoted: $v_1 \overset{\delta}{\rightsquigarrow} v_2$. Likewise the total cost of a path from a node in set 2 to a node in set 1 via a node in set p with a total cost of δ shall be written: $v_2 \overset{\omega_1}{\rightsquigarrow} v_p \overset{\omega_2}{\rightsquigarrow} v_1 = \delta$, where $\delta = \omega_1 + \omega_2$.

The first constraint, that parallel paths do not cross, can be enforced by ensuring that:

PARALLEL_1: $v_1 \rightsquigarrow v_p = \infty$

PARALLEL_2: $v_2 \rightsquigarrow v_p = \infty$

PARALLEL_3: $v_1 \rightsquigarrow v_2 = \infty$

PARALLEL_4: $v_2 \rightsquigarrow v_1 = \infty$

PARALLEL_5: $s \rightarrow v_p = \infty$

We can impose the parallel constraints by removing the last edge from every possible path that contradicts these constraints, thereby giving these paths an infinite cost eg. $v_1 \rightsquigarrow v_p = \infty \equiv v_1 \rightsquigarrow v_1 \nrightarrow v_p$. Put simply, this means we disconnect all incoming edges from the set of nodes on the path. However the notation is useful to talk about the second set of constraints.

The constraint that orthogonal paths cross only once can be enforced by ensuring that:

ORTHOGONAL_1: $v_1 \rightsquigarrow v_2 \geq \gamma$

ORTHOGONAL_2: $v_1 \rightsquigarrow v_p \rightsquigarrow v_2 \geq \gamma$

ORTHOGONAL_3: $v_p \rightsquigarrow v_p \geq \gamma$

ORTHOGONAL_4: $v_2 \rightsquigarrow v_p = \infty$

ORTHOGONAL_5: $v_2 \rightsquigarrow v_1 = \infty$

A method for imposing these constraints on sets of edges will depend on the particular graph construction - so we deal with each in turn.

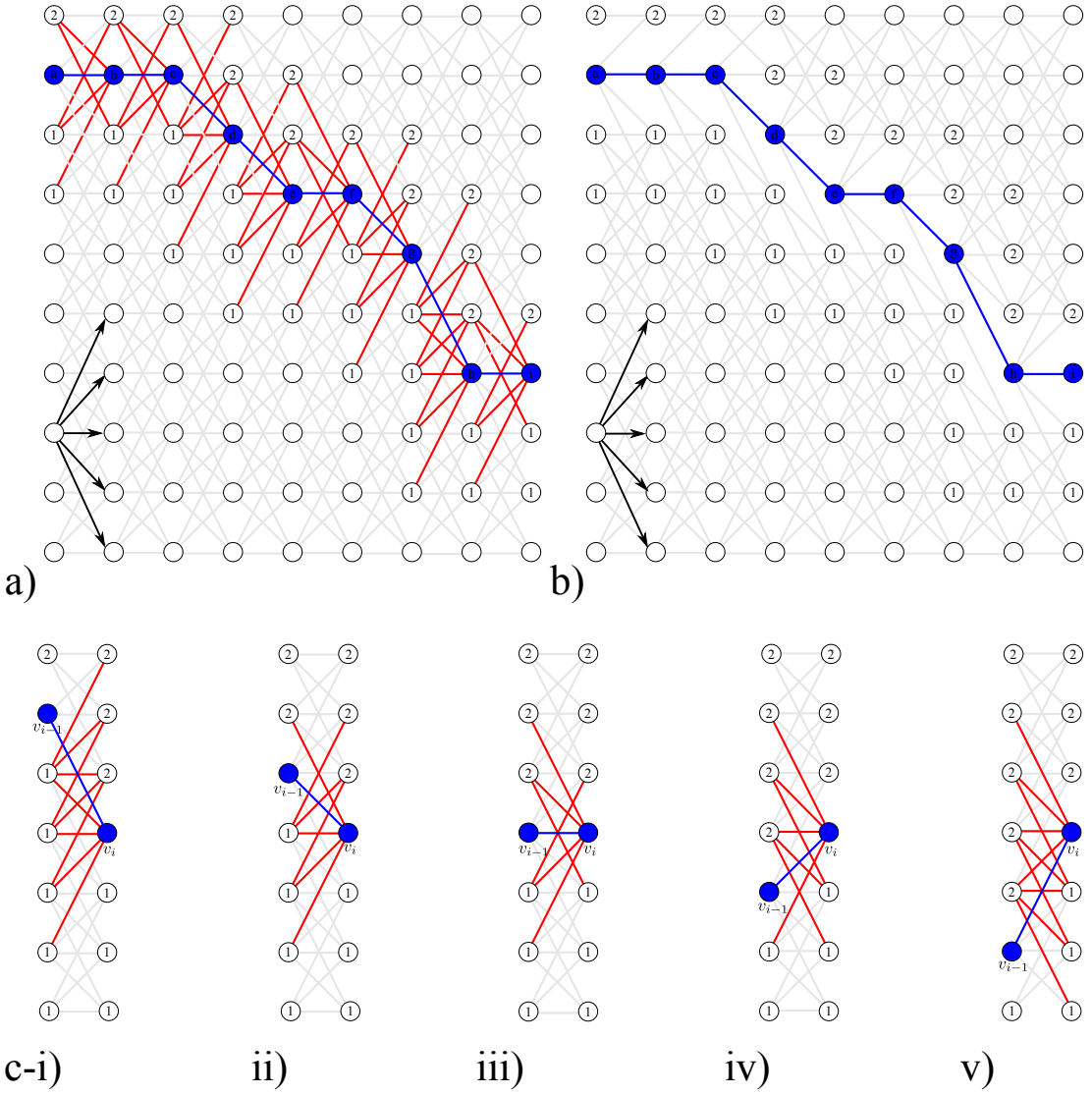


Figure A.2: Parallel path constraints for G_{dag}. a) Example constraint edges for a parallel path in a horizontal strip. The edge connectivity is shown in black, bottom left hand corner. Removing this set of edges prevent possible paths from sets of vertices 1 and 2. b) This set of edges is easier to view if they are removed from the graph. Additionally, in practice (when using shortest-paths algorithms) removing edges and adding constraint edges will have the same effect on the solutions as these edges will not appear in the final solution. So in an implementation there is only need for one edge function. c) Set of predecessor edges for 5-connected dag.

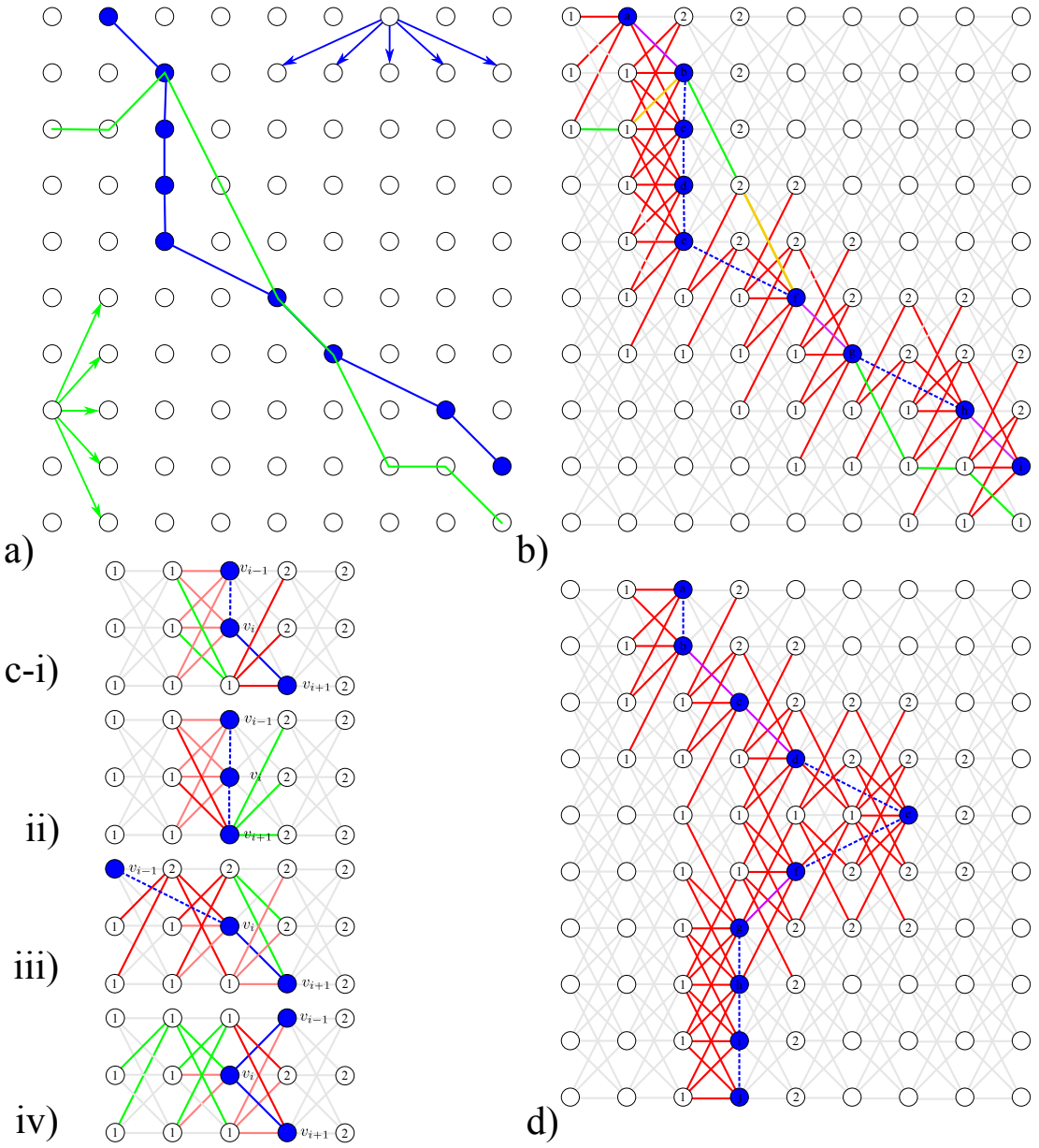


Figure A.3: Orthogonal path constraints for G_{dag}. a) Two legal paths in orthogonal directions that can overlap multiple times. To prevent this we require additional edge constraints. b) Example edge constraints, for the green horizontal path crossing the blue vertical path, are shown in red. Purple edges occur in both horizontal and vertical directions. Yellow edges on original green path now occur additional large cost. The cost of the original green path is now increase by 4γ and will not appear in the final shortest path for this horizontal strip. c) Examples where it is impossible to decide which edges are constraint edges if we only consider a node and its predecessor. Comparing i-ii the pink edges are constant but the red and green edges vary depending on v_{i+1}. Dotted blue edges are on the path in the orthogonal graph that have no equivalent edge in this graph. d) A further example of using these constraints on a path marked by blue nodes.

A.1 Constraints for G_{dag}

For this graph construction the sets of edges at a given node are not identical in the horizontal and vertical directions. Moreover there are edges between nodes that are not nearest neighbours in Euclidean distance.

We could naively impose the parallel constraints by visiting every node and testing the edge properties against the five constraint criteria. However, the restriction to non-returning paths in G_{dag} means there is a linear ordering and nodes can be treated independently based on their predecessor. This means we need only visit nodes based along the shortest path $p = \langle \nu_k, \nu_k - 1, \ldots, \nu_0 \rangle$ using the predecessor attribute, $\pi[v]$.

Based on the relative position of v_{i-1} to node v_i we then remove the set of edges using the five parallel constraints. This is illustrated in Figure A.2. The set of edges removed based on predecessor orientations for a 5-connected dag can be seen in Figure A.2c. This book keeping operation requires visiting a number of nodes dependent on the shape of the shortest-path, eg. fewer edges are removed for straight horizontal paths. However an upper bound for this, based on the graph connectivity, is linear in the number of nodes in the path and it therefore does not increase the complexity of $\mathcal{O}(V + E)$.

The linear ordering of nodes in the dag is not sufficient to prevent paths in orthogonal directions overlapping multiple times. This is illustrated in Figure A.3a. We must therefore alter the edges along existing paths to impose the orthogonal constraints.

ORTHOGONAL_1 can be imposed by setting the connecting edge for set 1 to set 2 for each path to the constant γ: $v_1 \rightsquigarrow v_2 \geq \gamma \equiv v_1 \rightsquigarrow v_1 \xrightarrow{\gamma} v_2 \rightsquigarrow v_2$. ORTHOGONAL_4 and ORTHOGONAL_5 are set in a similar manner. To set ORTHOGONAL_3 we need to find every edge in the constraint path that is also present in this graph and set it to γ. These edges are highlighted in purple in Figure A.3b.

However there is a choice to be made in how to apply ORTHOGONAL_2 as either the connecting edge of the path $v_1 \rightsquigarrow v_p$, or connecting edge of the path $v_p \rightsquigarrow v_2$, or a combination of both can be altered.

One possible update scheme that achieves this is illustrated in Figure A.3b. This corresponds to the choice of altering the first edge incident on the path node: $v_1 \rightsquigarrow v_1 \xrightarrow{\gamma} v_p \rightsquigarrow v_2 \geq \gamma$. This scheme means that the cost of our original horizontal path, green, is now $v_1 \rightsquigarrow v_1 \xrightarrow{\gamma} v_p \rightsquigarrow v_2 \geq 4\gamma$ where the three visible edges that are altered

are $1-b$, $2-f$ and $f-g$. This means that this can no longer be a shortest path from s to t in this graph.

Again we could naively impose these constraints by visiting every node but we can reduce this considerably. However, the orthogonal constraints are being imposed using a path from a graph with an alternative edge structure which means that possible paths from one set of nodes to the next are no longer fully specified by a single predecessor. We need to consider a *template* of possible paths based on the connectivity in the orthogonal dag. An example of this is illustrated in Figure A.3c. If we compare Figure A.3c-i-ii we can see that if we only consider nodes (v_i, v_{i-1}) we cannot tell which of the edges highlighted in red or green need to be altered. Similarly if we compare Figure A.3c-iii-iv we again cannot decide on the highlighted edges if we only consider nodes (v_i, v_{i+1}). Therefore, when updating the graph edges we must consider templates based on sets of predecessors. The size of the set of predecessors is governed by the connectivity of the graph used in the orthogonal image strip. For the 5-connected dag there are a possible 25 permutations of these constraint templates. An additional example of updating a path is given in Figure A.3d.

Again this book keeping requires visiting a number of nodes dependent on the shape of the shortest-path but an upper bound based on fixed connectivity is linear in the number of nodes in the path and it therefore does not increase the complexity of $\mathcal{O}(V+E)$.

A.2 Constraints for G_{gg+}

For this graph construction the sets of edges at a given node are identical for strips in the horizontal and vertical directions and are only between nodes that are nearest neighbours in Euclidean distance. This makes enforcing the parallel constraints easy as we need only remove edges that are incident to nodes on the path. Additionally, the orthogonal constraints can be imposed by only visiting nodes neighbouring the path.

The ORTHOGONAL_1 and ORTHOGONAL_5 constraints do not apply in the grid graph as there are no paths from set 1 to set 2 that do not pass through set p. ORTHOGONAL_3 is imposed by setting every link between path nodes to a constant. To do this we must proceed along the predecessor chain using the attribute $\pi[v]$ and set a flag so that we know whether each node is in p. Note that this time the nodes belonging

to p that are adjacent need not be predecessors on that path. However the cost of the constant used in ORTHOGONAL_3 depends on our solution to the ORTHOGONAL_2.

To set ORTHOGONAL_2 and ORTHOGONAL_4 and we need to know which neighbouring nodes are in set 1 and set 2. We can achieve this by proceeding down the path taking into account the region of neighbouring nodes and additionally the direction of the predecessor (v_i,v_{i-1}) - so that we know whether nodes are on the 'right' or 'left' of the path as we proceed along the chain. An example of this is illustrated in Figure A.4a. As we proceed back down the chain 4-connected nodes that are on the 'right' of our direction of travel, that are not on the path, are in set 2, vice versa for set 1. We can therefore set ORTHOGONAL_4 by altering the edges: $v_2 \rightarrow v_p = \gamma$.

As in the previous graph our choice of how to implement ORTHOGONAL_2 depends on how we alter $v_1 \rightsquigarrow v_p$, $v_p \rightsquigarrow v_2$, or both. However, we cannot use the same simple scheme as the previous graph because it has the undesirable consequence of biasing paths to the metrication effects of the grid graph. For example, if we always apply $v_1 \rightsquigarrow v_1 \overset{\gamma}{\rightarrow} v_p \rightsquigarrow v_2 \geq \gamma$ then the result is that $v_1 \overset{\gamma}{\rightarrow} v_p \rightarrow v_2 < v_1 \overset{\gamma}{\rightarrow} v_p \overset{\omega}{\rightsquigarrow} v_p \rightsquigarrow v_2$, where we require that $\omega > 0$ so that paths cannot follow the same boundaries. This would mean there is a lower cost to cross an existing path at straight regions rather than say diagonal or curved regions. Furthermore, because the cost of the constant is dominant it can mean that a path deviates wildly across a particular strip of an image so that it no longer captures the original boundary information that remains in the graph. To avoid this we must alter how the constant cost is accrued across the path depending on its local structure.

One possible way to do this is to fix the constant used for imposing ORTHOGONAL_3 so each edge in the set p costs γ: $v_p \overset{\gamma}{\rightarrow} v_p$. We must then alter the cost of the edge $v_1 \overset{\omega}{\rightarrow} v_p$ depending on whether is appears in the path $v_1 \overset{\omega+\delta}{\rightarrow} v_p \rightarrow v_2$ or $v_1 \overset{\omega+\delta}{\rightarrow} v_p \overset{\gamma}{\rightarrow} v_p \rightarrow v_2$. In the first path $\delta = \gamma$ and in the second $\delta = 0$. With this choice of edge scheme there are only 12 possible templates based on a set of three nodes on the path - the node, its successor and predecessor. These are illustrated in Figure A.4a where the templates are oriented so that the node v_{i-1} is always at the top. Examples of applying this scheme are give in Figure A.4c-d. For instance, in example A.4d we can see that the path 1-b-c-2 costs the same as the path 1-a-2 or the path 1-f-g-2.

Therefore, all the constraints can be updated in a single pass down the predeces-

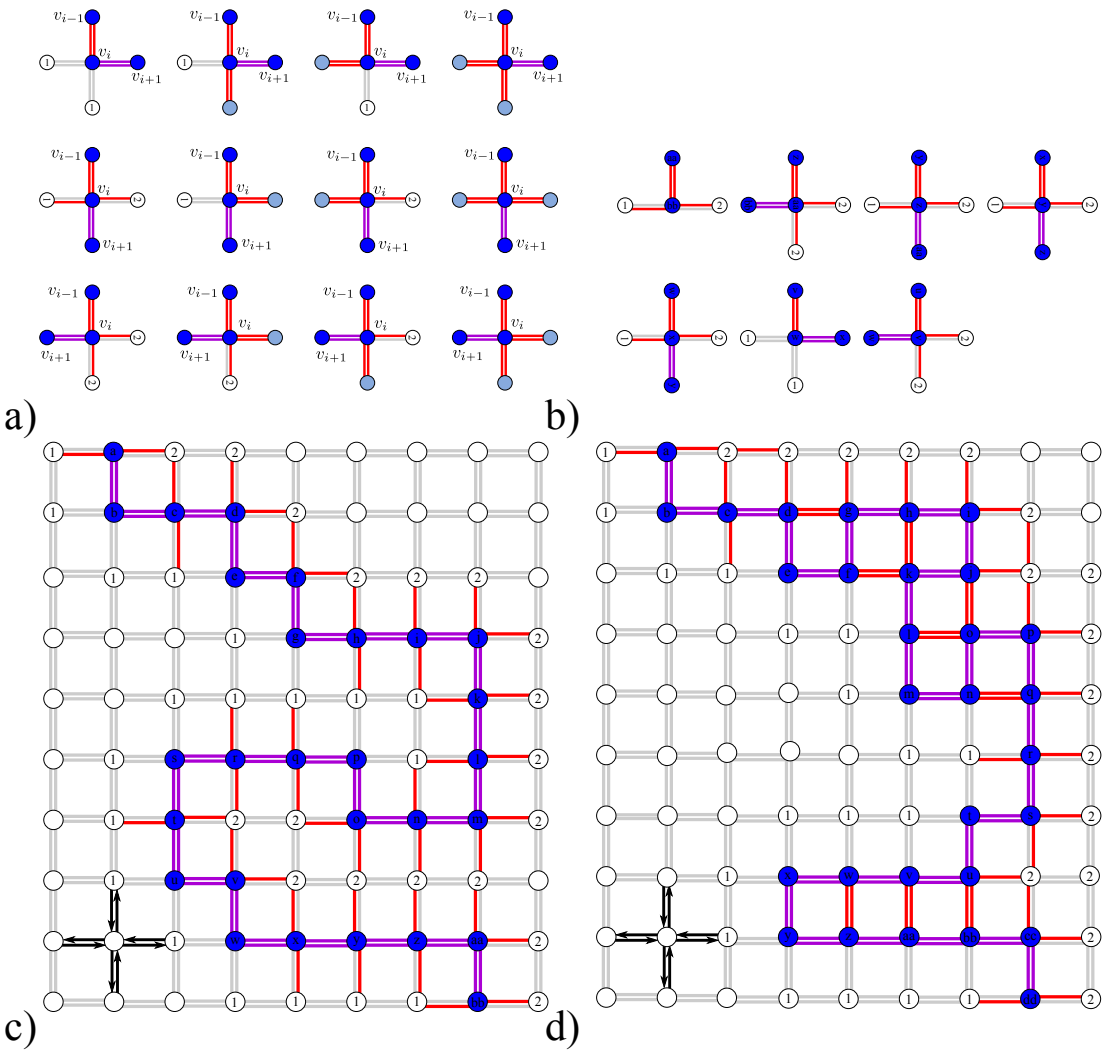


Figure A.4: Orthogonal path constraints for G_{++}. a) Set of 12 templates for nodes lying on the path, oriented so that the next predecessor, v_{i-1}, is always at the top. Nodes that are adjacent but are not in the immediate predecessor chain are in light blue. Edges between current node and its predecessor are purple and have the same cost as additional constraint edges, shown in red. All paths from set 1 to the nearest neighbour node in set 2 incur a cost γ b) Example edge updates as we proceed along the path in (c) - starting from the last node bb. c) An example of the orthogonal constraints applied to a path highlighted with blue nodes. d) A second example that contains adjacent path nodes that are not predecessors. Although in practice these paths occur less frequently because of the moderating effect of path length. These also incur the fixed cost and therefore 'block in' regions of continuous path to prevent spurious superpixels being generated.

sor chain. An illustration of this is shown in Figure A.4b.This book keeping is linear in the number of nodes in the path and it therefore does not increase the complexity of $\mathcal{O}(V \ln V + E)$. In practice we implement this update scheme using convolution because our boundary map data is stored in an array rather that a graph data structure.

Appendix B

Tables of Results

Not all results presented in the thesis will fit in one table on one page. Results are presented per chapter and repeated across tables where necessary to make comparisons easier.

B.1 Chapter 3. Berkeley Segmentation Database

Metric = Human	F_{sd} = 0.985						Cover Score = 0.955					
Lattice Resolution / Algorithm/ Parameters	42	100	210	342	600	900	42	100	210	342	600	900
	6 × 7	10 × 10	14 × 15	18 × 19	24 × 25	30 × 30	6 × 7	10 × 10	14 × 15	18 × 19	24 × 25	30 × 30
uniform	0.327	0.433	0.525	0.586	0.660	0.712	0.703	0.777	0.827	0.854	0.883	0.901
MS	0.407	0.577	0.711	0.754	0.798	0.834	0.684	0.795	0.880	0.903	0.918	0.933
FH	0.428	0.512	0.560	0.621	0.747	0.816	0.671	0.738	0.772	0.820	0.895	0.927
NC (Pb)	0.511	0.585	0.661	0.706	0.750	0.780	0.834	0.878	0.910	0.925	0.934	0.942
GRL (Canny)	0.415	0.521	0.623	0.678	0.732	0.773	0.711	0.791	0.851	0.878	0.902	0.916
GRL (Pb)	0.441	0.547	0.636	0.687	0.740	0.775	0.744	0.819	0.870	0.892	0.912	0.924
GRL (BEL)	0.451	0.555	0.645	0.698	0.747	0.782	0.760	0.828	0.875	0.897	0.915	0.925
GRL-MS(BEL)	0.464	0.581	0.675	0.725	0.772	0.800	0.767	0.845	0.887	0.904	0.919	0.924
GRL (ground truth)	0.501	0.604	0.693	0.741	0.794	0.824	0.854	0.904	0.929	0.939	0.947	0.952
	Prob RAND Index = 0.985						Global Consistency Error = 0.035					
uniform	0.883	0.916	0.938	0.948	0.959	0.966	0.228	0.190	0.155	0.135	0.111	0.095
MS	0.868	0.930	0.961	0.969	0.974	0.978	0.187	0.141	0.094	0.084	0.074	0.063
FH	0.842	0.887	0.904	0.934	0.966	0.977	0.112	0.116	0.124	0.122	0.089	0.069
NC (Pb)	0.945	0.960	0.970	0.975	0.978	0.981	0.121	0.099	0.080	0.070	0.062	0.056
GRL (Canny)	0.888	0.924	0.948	0.958	0.967	0.972	0.201	0.166	0.131	0.112	0.093	0.081
GRL (Pb)	0.906	0.936	0.956	0.964	0.970	0.974	0.176	0.143	0.114	0.099	0.083	0.073
GRL (BEL)	0.913	0.942	0.958	0.965	0.972	0.975	0.167	0.137	0.110	0.094	0.080	0.072
GRL-MS (BEL)	0.916	0.948	0.963	0.968	0.974	0.975	0.160	0.127	0.102	0.090	0.076	0.073
GRL (ground truth)	0.955	0.971	0.978	0.981	0.983	0.985	0.091	0.074	0.061	0.055	0.050	0.046
	Variation of Information = 0.335						Global Accuracy					
uniform	1.596	1.317	1.095	0.963	0.809	0.710	0.814	0.866	0.899	0.917	0.935	0.945
MeanShift	1.475	1.066	0.702	0.621	0.559	0.485	0.801	0.879	0.935	0.947	0.956	0.965
FH	1.289	1.101	1.036	0.928	0.656	0.521	0.777	0.831	0.858	0.896	0.942	0.961
NC (Pb)	0.921	0.752	0.608	0.539	0.486	0.446	0.903	0.931	0.951	0.959	0.965	0.969
GRL (Canny)	1.471	1.180	0.937	0.811	0.688	0.610	0.821	0.877	0.916	0.932	0.947	0.955
GRL (Pb)	1.306	1.033	0.828	0.729	0.626	0.562	0.844	0.895	0.927	0.940	0.952	0.959
GRL (BEL)	1.240	0.994	0.802	0.701	0.608	0.550	0.854	0.900	0.931	0.943	0.954	0.960
GRL-MS (BEL)	1.202	0.922	0.742	0.666	0.581	0.561	0.859	0.911	0.937	0.948	0.957	0.960
GRL (ground truth)	0.759	0.590	0.484	0.440	0.404	0.376	0.912	0.945	0.960	0.967	0.971	0.974

Table B.1: Performance measures on BSD comparing GRL (G_{gg+}) against Mean Shift [53], Minimum Spanning Tree (FH) [88] and Normalized Cuts (NC) [185]. Results are displayed for six different lattice resolutions and three different boundary maps. The human score - ie. agreement between different human subjects - is given in the header by each measure.

B.2 Chapter 3. Pascal Visual Object Classes

Metric = Human mode	F_{sd}						Cover Score					
Lattice Resolution / Algorithm/ Parameters	42	100	210	342	600	900	42	100	210	342	600	900
	6 × 7	10 × 10	14 × 15	18 × 19	24 × 25	30 × 30	6 × 7	10 × 10	14 × 15	18 × 19	24 × 25	30 × 30
uniform	0.635	0.739	0.816	0.851	0.874	0.886	0.581	0.639	0.699	0.736	0.778	0.806
MS	0.498	0.684	0.782	0.806	0.830	0.867	0.644	0.732	0.805	0.827	0.847	0.863
FH	0.613	0.660	0.695	0.730	0.786	0.848	0.676	0.694	0.726	0.759	0.803	0.847
NC (Pb)	0.680	0.742	0.786	0.820	0.849	0.866	0.717	0.761	0.806	0.827	0.842	0.855
GRL (Canny)	0.629	0.752	0.832	0.865	0.891	0.900	0.618	0.677	0.738	0.772	0.809	0.832
GRL (Pb)	0.629	0.706	0.812	0.877	0.897	0.911	0.654	0.695	0.755	0.795	0.824	0.843
GRL (BEL)	0.674	0.780	0.854	0.880	0.897	0.908	0.652	0.710	0.766	0.796	0.824	0.845
GRL-MS (BEL)	0.711	0.828	0.872	0.885	0.898	0.903	0.643	0.724	0.785	0.815	0.841	0.851
GRL (ground truth)	0.776	0.865	0.904	0.921	0.936	0.944	0.787	0.832	0.870	0.887	0.903	0.927
	RAND Index						Global Consistency Error					
uniform	0.649	0.694	0.743	0.775	0.810	0.835	0.226	0.214	0.192	0.173	0.149	0.132
MS	0.680	0.766	0.839	0.863	0.886	0.902	0.146	0.123	0.095	0.089	0.081	0.072
FH	0.703	0.733	0.773	0.803	0.843	0.884	0.109	0.124	0.129	0.119	0.103	0.085
NC (Pb)	0.792	0.828	0.864	0.882	0.893	0.903	0.157	0.143	0.122	0.109	0.099	0.093
GRL (Canny)	0.676	0.726	0.780	0.809	0.841	0.862	0.185	0.179	0.159	0.142	0.121	0.108
GRL (Pb)	0.697	0.741	0.795	0.830	0.855	0.873	0.149	0.153	0.140	0.124	0.109	0.097
GRL (BEL)	0.704	0.756	0.806	0.833	0.859	0.876	0.165	0.158	0.138	0.123	0.107	0.096
GRL-MS (BEL)	0.714	0.777	0.829	0.859	0.882	0.892	0.165	0.145	0.123	0.109	0.096	0.093
GRL (ground truth)	0.823	0.862	0.896	0.910	0.921	0.927	0.098	0.091	0.080	0.073	0.066	0.062
	Variation of Information						Global Accuracy					
uniform	1.349	1.245	1.108	1.100	0.891	0.804	0.916	0.939	0.956	0.964	0.971	0.976
MS	1.131	0.935	0.755	0.699	0.640	0.582	0.912	0.957	0.973	0.977	0.980	0.983
FH	0.999	1.005	0.964	0.884	0.765	0.636	0.923	0.940	0.953	0.964	0.974	0.982
NC (Pb)	1.049	0.934	0.805	0.737	0.690	0.652	0.956	0.966	0.975	0.979	0.982	0.983
GRL (Canny)	1.223	1.118	0.976	0.883	0.778	0.707	0.922	0.947	0.964	0.970	0.76	0.980
GRL (Pb)	1.119	1.056	0.921	0.815	0.729	0.669	0.929	0.948	0.966	0.974	0.978	0.981
GRL (BEL)	1.137	1.024	0.889	0.807	0.720	0.661	0.935	0.955	0.969	0.974	0.979	0.981
GRL-MS (BEL)	1.108	0.950	0.811	0.734	0.660	0.638	0.943	0.964	0.973	0.977	0.980	0.980
GRL (ground truth)	0.787	0.382	0.583	0.533	0.480	0.445	0.970	0.979	0.984	0.987	0.988	0.990

Table B.2: Performance measures on VOC.

B.3 Chapter 3. Cambridge Labeled Video Database

Metric = Human mode	F_{sd}						Cover Score					
Lattice Resolution / Algorithm/ Parameters	42	100	210	342	600	900	42	100	210	342	600	900
	6 × 7	10 × 10	14 × 15	18 × 19	24 × 25	30 × 30	6 × 7	10 × 10	14 × 15	18 × 19	24 × 25	30 × 30
uniform	0.282	0.329	0.381	0.429	0.483	0.527	0.630	0.693	0.743	0.772	0.798	0.817
MS	0.217	0.280	0.396	0.493	0.568	0.623	0.477	0.567	0.687	0.761	0.801	0.831
FH	0.242	0.373	0.445	0.496	0.534	0.550	0.439	0.654	0.715	0.746	0.767	0.776
NC (Pb)	0.317	0.361	0.423	0.466	0.512	0.549	0.741	0.776	0.808	0.824	0.838	0.848
GRL (Canny)	0.215	0.268	0.337	0.391	0.465	0.517	0.608	0.699	0.758	0.786	0.815	0.833
GRL (Pb)	0.202	0.259	0.326	0.375	0.447	0.4957	0.643	0.706	0.757	0.783	0.809	0.826
GRL (BEL)	0.280	0.342	0.411	0.462	0.529	0.567	0.651	0.720	0.772	0.799	0.824	0.841
GRL-MS (BEL)	0.300	0.379	0.464	0.525	0.580	0.591	0.658	0.742	0.789	0.816	0.837	0.842
GRL (ground truth)	0.243	0.308	0.388	0.454	0.532	0.590	0.715	0.770	0.816	0.839	0.862	0.876
	RAND Index						Global Consistency Error					
uniform	0.871	0.902	0.925	0.935	0.945	0.952	0.267	0.222	0.183	0.161	0.141	0.128
MS	0.783	0.836	0.898	0.930	0.945	0.956	0.273	0.246	0.192	0.157	0.139	0.121
FH	0.774	0.874	0.907	0.923	0.932	0.935	0.148	0.120	0.179	0.163	0.155	0.151
NC (Pb)	0.924	0.938	0.950	0.955	0.960	0.963	0.162	0.150	0.135	0.126	0.118	0.111
GRL (Canny)	0.869	0.905	0.929	0.941	0.951	0.957	0.240	0.210	0.172	0.151	0.131	0.118
GRL (Pb)	0.879	0.909	0.931	0.941	0.950	0.955	0.230	0.200	0.167	0.149	0.133	0.121
GRL (BEL)	0.882	0.915	0.936	0.946	0.955	0.960	0.227	0.194	0.161	0.144	0.125	0.114
GRL-MS (BEL)	0.884	0.923	0.941	0.951	0.959	0.961	0.233	0.186	0.154	0.134	0.118	0.113
GRL (ground truth)	0.912	0.934	0.952	0.960	0.967	0.971	0.176	0.149	0.123	0.109	0.094	0.084
	Variation of Information						Global Accuracy					
uniform	1.901	1.620	1.378	1.238	1.099	1.003	0.774	0.819	0.855	0.874	0.890	0.902
MS	2.115	1.869	1.448	1.158	0.933	0.868	0.654	0.723	0.814	0.867	0.892	0.912
FH	2.108	1.581	1.366	1.232	1.159	1.120	0.590	0.784	0.828	0.853	0.868	0.876
NC (Pb)	1.270	1.147	1..020	0.953	0.892	0.844	0.850	0.874	0.895	0.906	0.915	0.921
GRL (Canny)	1.852	1.547	1.290	1.152	1.010	0.915	0.761	0.823	0.864	0.883	0.901	0.913
GRL (Pb)	1.724	1.492	1.270	1.148	1.025	0.942	0.781	0.827	0.863	0.881	0.898	0.909
GRL (BEL)	1.695	1.437	1.209	1.084	0.955	0.871	0.788	0.836	0.872	0.890	0.907	0.918
GRL-MS (BEL)	1.691	1.354	1.141	0.998	0.894	0.867	0.792	0.851	0.883	0.902	0.915	0.918
GRL (ground truth)	1.363	1.155	0.958	0.851	0.739	0.667	0.832	0.871	0.902	0.917	0.931	0.939

Table B.3: Performance measures on CamVid.

B.4 Chapter 4. Cambridge Labeled Video Database

Metric = Human mode	F_{sd}						Cover Score					
Lattice Resolution / Algorithm/ Parameters	42	100	210	342	600	900	42	100	210	342	600	900
	6 × 7	10 × 10	14 × 15	18 × 19	24 × 25	30 × 30	6 × 7	10 × 10	14 × 15	18 × 19	24 × 25	30 × 30
uniform	0.282	0.329	0.381	0.429	0.483	0.527	0.630	0.693	0.743	0.772	0.798	0.817
MS	0.217	0.280	0.396	0.493	0.568	0.623	0.477	0.567	0.687	0.761	0.801	0.831
FH	0.242	0.373	0.445	0.496	0.534	0.550	0.439	0.654	0.715	0.746	0.767	0.776
NC (Pb)	0.317	0.361	0.423	0.466	0.512	0.549	0.741	0.776	0.808	0.824	0.838	0.848
GRL (BEL)	0.280	0.342	0.411	0.462	0.529	0.567	0.651	0.720	0.772	0.799	0.824	0.841
GRL-MS (BEL)	0.300	0.379	0.464	0.525	0.580	0.591	0.658	0.742	0.789	0.816	0.837	0.842
ARL-1(BEL)	0.289	0.356	0.437	0.482	0.545	0.588	0.687	0.749	0.799	0.824	0.848	0.862
ARL-4 (BEL)	0.290	0.356	0.438	0.484	0.546	0.588	0.689	0.750	0.800	0.824	0.849	0.863
ARL-MS (BEL)	0.324	0.404	0.488	0.542	0.590	0.624	0.730	0.784	0.823	0.844	0.863	0.870
	RAND Index						Global Consistency Error					
uniform	0.871	0.902	0.925	0.935	0.945	0.952	0.267	0.222	0.183	0.161	0.141	0.128
MS	0.783	0.836	0.898	0.930	0.945	0.956	0.273	0.246	0.192	0.157	0.139	0.121
FH	0.774	0.874	0.907	0.923	0.932	0.935	0.148	0.120	0.179	0.163	0.155	0.151
NC (Pb)	0.924	0.938	0.950	0.955	0.960	0.963	0.162	0.150	0.135	0.126	0.118	0.111
GRL (BEL)	0.882	0.915	0.936	0.946	0.955	0.960	0.227	0.194	0.161	0.144	0.125	0.114
GRL-MS (BEL)	0.884	0.923	0.941	0.951	0.959	0.961	0.233	0.186	0.154	0.134	0.118	0.113
ARL-1(BEL)	0.891	0.917	0.935	0.944	0.953	0.957	0.216	0.186	0.155	0.139	0.122	0.112
ARL-4 (BEL)	0.892	0.918	0.938	0.946	0.954	0.959	0.215	0.185	0.155	0.139	0.121	0.111
ARL-MS (BEL)	0.909	0.930	0.945	0.952	0.958	0.961	0.182	0.160	0.141	0.128	0.114	0.109
	Variation of Information						Global Accuracy					
uniform	1.901	1.620	1.378	1.238	1.099	1.003	0.774	0.819	0.855	0.874	0.890	0.902
MS	2.115	1.869	1.448	1.158	0.933	0.868	0.654	0.723	0.814	0.867	0.892	0.912
FH	2.108	1.581	1.366	1.232	1.159	1.120	0.590	0.784	0.828	0.853	0.868	0.876
NC (Pb)	1.270	1.147	1..020	0.953	0.892	0.844	0.850	0.874	0.895	0.906	0.915	0.921
GRL (BEL)	1.695	1.437	1.209	1.084	0.955	0.871	0.788	0.836	0.872	0.890	0.907	0.918
GRL-MS (BEL)	1.691	1.354	1.141	0.998	0.894	0.867	0.792	0.851	0.883	0.902	0.915	0.918
ARL-1(BEL)	1.620	1.391	1.182	1.068	0.948	0.875	0.796	0.838	0.873	0.889	0.906	0.915
ARL-4 (BEL)	1.618	1.390	1.180	1.067	0.947	0.873	0.797	0.840	0.874	0.890	0.906	0.915
ARL-MS (BEL)	1.394	1.208	1.058	0.970	0.876	0.846	0.825	0.862	0.889	0.902	0.914	0.918

Table B.4: Performance measures on CamVid.

B.5 Chapter 4. Cambridge Labeled Video Database

Algorithm	ARL (BEL)						ARL-[MS] (BEL)					
Mean Lattice Resolution / Selected Classes	~42	~100	~210	~342	~600	~900	~42	~100	~210	~342	~600	~900
	6 × 7	10 × 10	14 × 15	18 × 19	24 × 25	30 × 30	6 × 7	10 × 10	14 × 15	18 × 19	24 × 25	30 × 30
Bicyclist	0.049	0.117	0.196	0.279	0.380	0.463	0.053	0.138	0.228	0.364	0.473	0.516
Building	0.447	0.497	0.558	0.586	0.613	0.636	0.456	0.522	0.568	0.607	0.636	0.641
Car	0.185	0.339	0.527	0.648	0.721	0.769	0.220	0.411	0.575	0.698	0.767	0.796
Column/Pole	0.014	0.040	0.093	0.130	0.182	0.230	0.019	0.062	0.107	0.169	0.233	0.264
Fence	0.384	0.502	0.613	0.664	0.729	0.763	0.412	0.551	0.642	0.719	0.756	0.752
Pedestrian	0.041	0.095	0.192	0.274	0.381	0.451	0.053	0.145	0.234	0.344	0.456	0.503
Road	0.529	0.543	0.567	0.585	0.592	0.608	0.535	0.555	0.571	0.587	0.601	0.605
Sidewalk	0.550	0.652	0.729	0.768	0.805	0.835	0.563	0.666	0.731	0.782	0.817	0.830
Sign/Symbol	0.020	0.059	0.125	0.228	0.354	0.442	0.031	0.085	0.177	0.313	0.456	0.532
Sky	0.549	0.590	0.630	0.655	0.672	0.681	0.557	0.608	0.640	0.659	0.681	0.690
Tree	0.294	0.399	0.473	0.522	0.566	0.589	0.322	0.427	0.494	0.545	0.579	0.593
	GRL (BEL)						GRL-[MS] (BEL)					
Bicyclist	0.036	0.119	0.203	0.277	0.338	0.428	0.073	0.194	0.266	0.377	0.456	0.478
Building	0.432	0.483	0.547	0.582	0.614	0.636	0.468	0.536	0.576	0.621	0.637	0.641
Car	0.157	0.278	0.408	0.535	0.627	0.694	0.197	0.386	0.512	0.647	0.722	0.749
Column/Pole	0.019	0.049	0.097	0.139	0.193	0.242	0.031	0.088	0.138	0.211	0.269	0.281
Fence	0.353	0.488	0.571	0.625	0.697	0.739	0.414	0.554	0.627	0.711	0.740	0.750
Pedestrian	0.026	0.071	0.150	0.218	0.316	0.395	0.048	0.142	0.227	0.355	0.440	0.459
Road	0.526	0.547	0.568	0.578	0.589	0.602	0.542	0.559	0.567	0.596	0.597	0.598
Sidewalk	0.492	0.608	0.691	0.735	0.774	0.805	0.517	0.649	0.697	0.750	0.789	0.807
Sign/Symbol	0.015	0.036	0.084	0.169	0.293	0.364	0.025	0.092	0.176	0.307	0.434	0.472
Sky	0.549	0.594	0.640	0.660	0.680	0.691	0.572	0.629	0.658	0.683	0.691	0.693
Tree	0.301	0.392	0.466	0.509	0.553	0.584	0.346	0.455	0.505	0.559	0.588	0.593
	FH						MS					
Bicyclist	0.107	0.138	0.201	0.257	0.296	0.325	0.020	0.033	0.093	0.256	0.333	0.438
Building	0.489	0.512	0.536	0.556	0.559	0.560	0.422	0.463	0.526	0.582	0.613	0.643
Car	0.305	0.406	0.498	0.553	0.580	0.604	0.093	0.158	0.311	0.552	0.643	0.716
Column/Pole	0.124	0.175	0.264	0.330	0.365	0.404	0.027	0.056	0.151	0.334	0.414	0.482
Fence	0.315	0.340	0.388	0.440	0.479	0.525	0.339	0.388	0.501	0.561	0.595	0.668
Pedestrian	0.063	0.118	0.191	0.242	0.278	0.338	0.058	0.106	0.238	0.464	0.564	0.626
Road	0.543	0.549	0.568	0.600	0.614	0.616	0.479	0.481	0.489	0.528	0.566	0.600
Sidewalk	0.314	0.416	0.506	0.563	0.592	0.612	0.326	0.377	0.476	0.590	0.626	0.664
Sign/Symbol	0.019	0.259	0.371	0.456	0.496	0.538	0.054	0.113	0.251	0.459	0.533	0.610
Sky	0.723	0.768	0.796	0.803	0.803	0.809	0.425	0.494	0.629	0.715	0.736	0.751
Tree	0.349	0.411	0.456	0.470	0.471	0.471	0.220	0.287	0.377	0.420	0.425	0.455
	NC (Pb)						uniform					
Bicyclist	0.056	0.113	0.201	0.226	0.322	0.402	0.015	0.083	0.162	0.217	0.287	0.378
Building	0.503	0.531	0.575	0.587	0.603	0.617	0.474	0.508	0.547	0.578	0.608	0.630
Car	0.212	0.282	0.418	0.477	0.608	0.673	0.146	0.217	0.297	0.404	0.549	0.641
Column/Pole	0.008	0.031	0.086	0.109	0.158	0.195	0.000	0.005	0.014	0.031	0.058	0.095
Fence	0.440	0.498	0.578	0.596	0.651	0.683	0.328	0.440	0.529	0.590	0.661	0.705
Pedestrian	0.060	0.115	0.225	0.272	0.373	0.439	0.013	0.053	0.120	0.199	0.294	0.366
Road	0.540	0.547	0.558	0.561	0.563	0.566	0.525	0.534	0.542	0.546	0.559	0.573
Sidewalk	0.600	0.633	0.680	0.691	0.707	0.722	0.500	0.607	0.693	0.727	0.754	0.785
Sign/Symbol	0.024	0.067	0.180	0.231	0.337	0.395	0.011	0.053	0.098	0.178	0.260	0.334
Sky	0.600	0.619	0.649	0.658	0.669	0.682	0.536	0.568	0.609	0.629	0.648	0.664
Tree	0.410	0.450	0.504	0.520	0.556	0.574	0.342	0.410	0.465	0.503	0.545	0.576

Table B.5: Performance measured using F_{sd} on CamVid for selected Classes.

B.6 Chapter 5. Berkeley Segmentation Database

Metric = Human	F_{sd} = 0.985						Cover Score = 0.955					
Lattice Resolution / Algorithm/ Parameters	42 (6 × 7)	100 (10 × 10)	210 (14 × 15)	342 (18 × 19)	600 (24 × 25)	900 (30 × 30)	42 (6 × 7)	100 (10 × 10)	210 (14 × 15)	342 (18 × 19)	600 (24 × 25)	900 (30 × 30)
uniform	0.327	0.433	0.525	0.586	0.660	0.712	0.703	0.777	0.827	0.854	0.883	0.901
MS	0.407	0.577	0.711	0.754	0.798	0.834	0.684	0.795	0.880	0.903	0.918	0.933
FH	0.428	0.512	0.560	0.621	0.747	0.816	0.671	0.738	0.772	0.820	0.895	0.927
NC (Pb)	0.511	0.585	0.661	0.706	0.750	0.780	0.834	0.878	0.910	0.925	0.934	0.942
GRL (BEL)	0.451	0.555	0.645	0.698	0.747	0.782	0.760	0.828	0.875	0.897	0.915	0.925
GRL-MS(BEL)	0.464	0.581	0.675	0.725	0.772	0.800	0.767	0.845	0.887	0.904	0.919	0.924
LC(Canny)	0.484	0.574	0.657	0.706	0.761	0.798	0.802	0.849	0.884	0.903	0.920	0.932
LC(Pb)	0.497	0.585	0.667	0.712	0.760	0.798	0.809	0.856	0.892	0.910	0.922	0.931
LC(BEL)	0.495	0.587	0.667	0.716	0.766	0.798	0.814	0.860	0.900	0.913	0.925	0.932
LC-MS(BEL)	0.511	0.612	0.697	0.741	0.781	0.807	0.826	0.875	0.907	0.918	0.922	0.926
	Prob RAND Index = 0.985						Global Consistency Error = 0.035					
uniform	0.883	0.916	0.938	0.948	0.959	0.966	0.228	0.190	0.155	0.135	0.111	0.095
MS	0.868	0.930	0.961	0.969	0.974	0.978	0.187	0.141	0.094	0.084	0.074	0.063
FH	0.842	0.887	0.904	0.934	0.966	0.977	0.112	0.116	0.124	0.122	0.089	0.069
NC (Pb)	0.945	0.960	0.970	0.975	0.978	0.981	0.121	0.099	0.080	0.070	0.062	0.056
GRL (BEL)	0.913	0.942	0.958	0.965	0.972	0.975	0.167	0.137	0.110	0.094	0.080	0.072
GRL-MS (BEL)	0.916	0.948	0.963	0.968	0.974	0.975	0.160	0.127	0.102	0.090	0.076	0.073
LC(Canny)	0.934	0.950	0.961	0.967	0.973	0.977	0.143	0.123	0.103	0.090	0.076	0.065
LC(Pb)	0.939	0.953	0.964	0.969	0.973	0.977	0.136	0.117	0.096	0.083	0.074	0.066
LC(BEL)	0.940	0.954	0.965	0.971	0.975	0.977	0.129	0.111	0.092	0.080	0.071	0.066
LC-MS(BEL)	0.946	0.962	0.973	0.976	0.978	0.980	0.122	0.102	0.082	0.074	0.070	0.066
	Variation of Information = 0.335						Global Accuracy					
uniform	1.596	1.317	1.095	0.963	0.809	0.710	0.814	0.866	0.899	0.917	0.935	0.945
MeanShift	1.475	1.066	0.702	0.621	0.559	0.485	0.801	0.879	0.935	0.947	0.956	0.965
FH	1.289	1.101	1.036	0.928	0.656	0.521	0.777	0.831	0.858	0.896	0.942	0.961
NC (Pb)	0.921	0.752	0.608	0.539	0.486	0.446	0.903	0.931	0.951	0.959	0.965	0.969
GRL (BEL)	1.240	0.994	0.802	0.701	0.608	0.550	0.854	0.900	0.931	0.943	0.954	0.960
GRL-MS (BEL)	1.202	0.922	0.742	0.666	0.581	0.561	0.859	0.911	0.937	0.948	0.957	0.960
LC(Canny)	1.067	0.903	0.759	0.671	0.578	0.511	0.884	0.914	0.9371	0.947	0.957	0.963
LC(Pb)	1.017	0.857	0.706	0.625	0.567	0.511	0.884	0.914	0.937	0.947	0.957	0.964
LC(BEL)	0.985	0.828	0.684	0.602	0.542	0.511	0.891	0.921	0.943	0.952	0.960	0.965
LC-MS(BEL)	0.955	0.779	0.634	0.564	0.527	0.503	0.901	0.933	0.952	0.958	0.960	0.963

Table B.6: Performance measures on BSD

B.7 Chapter 5. Pascal Visual Object Classes

Metric = Human mode	F_{sd}						Cover Score					
Lattice Resolution / Algorithm/ Parameters	42 (6 × 7)	100 (10 × 10)	210 (14 × 15)	342 (18 × 19)	600 (24 × 25)	900 (30 × 30)	42 (6 × 7)	100 (10 × 10)	210 (14 × 15)	342 (18 × 19)	600 (24 × 25)	900 (30 × 30)
uniform	0.635	0.739	0.816	0.851	0.874	0.886	0.581	0.639	0.699	0.736	0.778	0.806
MS	0.498	0.684	0.782	0.806	0.830	0.867	0.644	0.732	0.805	0.827	0.847	0.863
FH	0.613	0.660	0.695	0.730	0.786	0.848	0.676	0.694	0.726	0.759	0.803	0.847
NC (Pb)	0.680	0.742	0.786	0.820	0.849	0.866	0.717	0.761	0.806	0.827	0.842	0.855
GRL (BEL)	0.674	0.780	0.854	0.880	0.897	0.908	0.652	0.710	0.766	0.796	0.824	0.845
GRL-MS (BEL)	0.711	0.828	0.872	0.885	0.898	0.903	0.643	0.724	0.785	0.815	0.841	0.851
LC (BEL)	0.724	0.796	0.843	0.868	0.883	0.896	0.675	0.727	0.778	0.805	0.829	0.845
LC-MS (BEL)	0.740	0.816	0.860	0.875	0.887	0.898	0.679	0.742	0.795	0.822	0.844	0.850
	RAND Index						Global Consistency Error					
uniform	0.649	0.694	0.743	0.775	0.810	0.835	0.226	0.214	0.192	0.173	0.149	0.132
MS	0.680	0.766	0.839	0.863	0.886	0.902	0.146	0.123	0.095	0.089	0.081	0.072
FH	0.703	0.733	0.773	0.803	0.843	0.884	0.109	0.124	0.129	0.119	0.103	0.085
NC (Pb)	0.792	0.828	0.864	0.882	0.893	0.903	0.157	0.143	0.122	0.109	0.099	0.093
GRL (BEL)	0.704	0.756	0.806	0.833	0.859	0.876	0.165	0.158	0.138	0.123	0.107	0.096
GRL-MS (BEL)	0.714	0.777	0.829	0.859	0.882	0.892	0.165	0.145	0.123	0.109	0.096	0.093
LC (BEL)	0.741	0.783	0.826	0.847	0.867	0.879	0.157	0.145	0.126	0.114	0.101	0.093
LC-MS (BEL)	0.741	0.791	0.837	0.861	0.883	0.892	0.155	0.137	0.117	0.105	0.094	0.093
	Variation of Information						Global Accuracy					
uniform	1.349	1.245	1.108	1.100	0.891	0.804	0.916	0.939	0.956	0.964	0.971	0.976
MS	1.131	0.935	0.755	0.699	0.640	0.582	0.912	0.957	0.973	0.977	0.980	0.983
FH	0.999	1.005	0.964	0.884	0.765	0.636	0.923	0.940	0.953	0.964	0.974	0.982
NC (Pb)	1.049	0.934	0.805	0.737	0.690	0.652	0.956	0.966	0.975	0.979	0.982	0.983
GRL (BEL)	1.137	1.024	0.889	0.807	0.720	0.661	0.935	0.955	0.969	0.974	0.979	0.981
GRL-MS (BEL)	1.108	0.950	0.811	0.734	0.660	0.638	0.943	0.964	0.973	0.977	0.980	0.980
LC (BEL)	1.059	0.953	0.832	0.759	0.688	0.646	0.951	0.964	0.973	0.977	0.980	0.982
LC-MS (BEL)	1.045	0.909	0.783	0.712	0.651	0.641	0.954	0.968	0.976	0.979	0.980	0.981

Table B.7: Performance measures on VOC.

B.8 Chapter 5. Cambridge Labeled Video Database

Metric = Human mode	F_{sd}						Cover Score					
Lattice Resolution / Algorithm/ Parameters	42	100	210	342	600	900	42	100	210	342	600	900
	6 × 7	10 × 10	14 × 15	18 × 19	24 × 25	30 × 30	6 × 7	10 × 10	14 × 15	18 × 19	24 × 25	30 × 30
uniform	0.282	0.329	0.381	0.429	0.483	0.527	0.630	0.693	0.743	0.772	0.798	0.817
MS	0.217	0.280	0.396	0.493	0.568	0.623	0.477	0.567	0.687	0.761	0.801	0.831
FH	0.242	0.373	0.445	0.496	0.534	0.550	0.439	0.654	0.715	0.746	0.767	0.776
NC (Pb)	0.317	0.361	0.423	0.466	0.512	0.549	0.741	0.776	0.808	0.824	0.838	0.848
GRL (BEL)	0.280	0.342	0.411	0.462	0.529	0.567	0.651	0.720	0.772	0.799	0.824	0.841
GRL-MS (BEL)	0.300	0.379	0.464	0.525	0.580	0.591	0.658	0.742	0.789	0.816	0.837	0.842
ARL-4 (BEL)	0.290	0.356	0.438	0.484	0.546	0.588	0.689	0.750	0.800	0.824	0.849	0.863
ARL-MS (BEL)	0.299	0.373	0.460	0.518	0.581	0.624	0.697	0.762	0.811	0.834	0.861	0.870
ALC (BEL)	0.320	0.386	0.461	0.507	0.564	0.595	0.741	0.785	0.822	0.842	0.861	0.870
ALC-MS(BEL)	0.324	0.404	0.488	0.542	0.590	0.624	0.730	0.784	0.823	0.844	0.863	0.870
	RAND Index						Global Consistency Error					
uniform	0.871	0.902	0.925	0.935	0.945	0.952	0.267	0.222	0.183	0.161	0.141	0.128
MS	0.783	0.836	0.898	0.930	0.945	0.956	0.273	0.246	0.192	0.157	0.139	0.121
FH	0.774	0.874	0.907	0.923	0.932	0.935	0.148	0.120	0.179	0.163	0.155	0.151
NC (Pb)	0.924	0.938	0.950	0.955	0.960	0.963	0.162	0.150	0.135	0.126	0.118	0.111
GRL (BEL)	0.882	0.915	0.936	0.946	0.955	0.960	0.227	0.194	0.161	0.144	0.125	0.114
GRL-MS (BEL)	0.884	0.923	0.941	0.951	0.959	0.961	0.233	0.186	0.154	0.134	0.118	0.113
ARL-4 (BEL)	0.892	0.918	0.938	0.946	0.954	0.959	0.215	0.185	0.155	0.139	0.121	0.111
ARL-MS (BEL)	0.895	0.922	0.941	0.949	0.958	0.961	0.211	0.175	0.146	0.130	0.113	0.109
ALC-4 (BEL)	0.914	0.932	0.945	0.952	0.958	0.961	0.172	0.153	0.136	0.125	0.113	0.107
ALC-MS(BEL)	0.909	0.930	0.945	0.952	0.958	0.961	0.182	0.160	0.141	0.128	0.114	0.109
	Variation of Information						Global Accuracy					
uniform	1.901	1.620	1.378	1.238	1.099	1.003	0.774	0.819	0.855	0.874	0.890	0.902
MS	2.115	1.869	1.448	1.158	0.933	0.868	0.654	0.723	0.814	0.867	0.892	0.912
FH	2.108	1.581	1.366	1.232	1.159	1.120	0.590	0.784	0.828	0.853	0.868	0.876
NC (Pb)	1.270	1.147	1..020	0.953	0.892	0.844	0.850	0.874	0.895	0.906	0.915	0.921
GRL (BEL)	1.695	1.437	1.209	1.084	0.955	0.871	0.788	0.836	0.872	0.890	0.907	0.918
GRL-MS (BEL)	1.691	1.354	1.141	0.998	0.894	0.867	0.792	0.851	0.883	0.902	0.915	0.918
ARL-4 (BEL)	1.618	1.390	1.180	1.067	0.947	0.873	0.797	0.840	0.874	0.890	0.906	0.915
ARL-MS (BEL)	1.586	1.332	1.122	1.007	0.883	0.846	0.802	0.848	0.881	0.897	0.913	0.918
ALC-4 (BEL)	1.345	1.192	1.053	0.968	0.879	0.838	0.832	0.862	0.887	0.901	0.913	0.918
ALC-MS(BEL)	1.394	1.208	1.058	0.970	0.876	0.846	0.825	0.862	0.889	0.902	0.914	0.918

Table B.8: Performance measures on CamVid.

B.9 Chapter 5. Cambridge Labeled Video Database

Algorithm	ARL-4 (BEL)						ARL-[MS] (BEL)					
Mean Lattice Resolution / Selected Classes	~42	~100	~210	~342	~600	~900	~42	~100	~210	~342	~600	~900
	6 × 7	10 × 10	14 × 15	18 × 19	24 × 25	30 × 30	6 × 7	10 × 10	14 × 15	18 × 19	24 × 25	30 × 30
Bicyclist	0.049	0.117	0.196	0.279	0.380	0.463	0.053	0.138	0.228	0.364	0.473	0.516
Building	0.447	0.497	0.558	0.586	0.613	0.636	0.456	0.522	0.568	0.607	0.636	0.641
Car	0.185	0.339	0.527	0.648	0.721	0.769	0.220	0.411	0.575	0.698	0.767	0.796
Column/Pole	0.014	0.040	0.093	0.130	0.182	0.230	0.019	0.062	0.107	0.169	0.233	0.264
Fence	0.384	0.502	0.613	0.664	0.729	0.763	0.412	0.551	0.642	0.719	0.756	0.752
Pedestrian	0.041	0.095	0.192	0.274	0.381	0.451	0.053	0.145	0.234	0.344	0.456	0.503
Road	0.529	0.543	0.567	0.585	0.592	0.608	0.535	0.555	0.571	0.587	0.601	0.605
Sidewalk	0.550	0.652	0.729	0.768	0.805	0.835	0.563	0.666	0.731	0.782	0.817	0.830
Sign/Symbol	0.020	0.059	0.125	0.228	0.354	0.442	0.031	0.085	0.177	0.313	0.456	0.532
Sky	0.549	0.590	0.630	0.655	0.672	0.681	0.557	0.608	0.640	0.659	0.681	0.690
Tree	0.294	0.399	0.473	0.522	0.566	0.589	0.322	0.427	0.494	0.545	0.579	0.593
	GRL (BEL)						GRL-[MS] (BEL)					
Bicyclist	0.036	0.119	0.203	0.277	0.338	0.428	0.073	0.194	0.266	0.377	0.456	0.478
Building	0.432	0.483	0.547	0.582	0.614	0.636	0.468	0.536	0.576	0.621	0.637	0.641
Car	0.157	0.278	0.408	0.535	0.627	0.694	0.197	0.386	0.512	0.647	0.722	0.749
Column/Pole	0.019	0.049	0.097	0.139	0.193	0.242	0.031	0.088	0.138	0.211	0.269	0.281
Fence	0.353	0.488	0.571	0.625	0.697	0.739	0.414	0.554	0.627	0.711	0.740	0.750
Pedestrian	0.026	0.071	0.150	0.218	0.316	0.395	0.048	0.142	0.227	0.355	0.440	0.459
Road	0.526	0.547	0.568	0.578	0.589	0.602	0.542	0.559	0.567	0.596	0.597	0.598
Sidewalk	0.492	0.608	0.691	0.735	0.774	0.805	0.517	0.649	0.697	0.750	0.789	0.807
Sign/Symbol	0.015	0.036	0.084	0.169	0.293	0.364	0.025	0.092	0.176	0.307	0.434	0.472
Sky	0.549	0.594	0.640	0.660	0.680	0.691	0.572	0.629	0.658	0.683	0.691	0.693
Tree	0.301	0.392	0.466	0.509	0.553	0.584	0.346	0.455	0.505	0.559	0.588	0.593
	ALC-4 (BEL)						ALC-[MS] (BEL)					
Bicyclist	0.063	0.135	0.244	0.307	0.378	0.472	0.064	0.174	0.243	0.355	0.469	0.531
Building	0.458	0.504	0.561	0.590	0.619	0.636	0.476	0.530	0.566	0.607	0.636	0.647
Car	0.253	0.411	0.559	0.659	0.730	0.774	0.282	0.485	0.592	0.710	0.768	0.800
Column/Pole	0.022	0.061	0.113	0.158	0.214	0.250	0.030	0.089	0.133	0.207	0.249	0.272
Fence	0.459	0.553	0.618	0.671	0.730	0.757	0.472	0.581	0.647	0.715	0.744	0.758
Pedestrian	0.058	0.141	0.240	0.304	0.395	0.459	0.079	0.192	0.273	0.390	0.467	0.512
Road	0.554	0.558	0.572	0.589	0.610	0.619	0.545	0.557	0.575	0.600	0.611	0.613
Sidewalk	0.592	0.687	0.744	0.774	0.805	0.834	0.572	0.681	0.718	0.761	0.804	0.832
Sign/Symbol	0.018	0.063	0.179	0.281	0.397	0.471	0.025	0.133	0.247	0.392	0.480	0.546
Sky	0.562	0.609	0.651	0.663	0.681	0.678	0.574	0.631	0.656	0.674	0.687	0.692
Tree	0.319	0.398	0.480	0.523	0.562	0.580	0.337	0.442	0.501	0.550	0.577	0.594

Table B.9: Performance measured using F_{sd} on CamVid for selected Classes.

Appendix C

Performance Measures

Evaluating a segmentation involves comparing one segmentation $\mathcal{S}$ to another $\mathcal{S}'$ where one of these segmentations can be considered an ideal, or close approximation, to the clustering in the real world - the ground truth. In this appendix we present the six region measures that are used in the quantitative evaluation in the thesis. In our experiments we first assign each superpixel to the mode class of the ground truth data before comparing segmentations. This can be interpreted as using an ideal classifier to label the set of pixels within a superpixel based on the dominant class and is similar to the notion of *potential accuracy* adopted in [84].

C.1 Region Measures

Most region based methods can be described using a *confusion matrix* for pixels belonging to two sets of regions, or segmentations $\mathcal{S}$ and $\mathcal{S}'$. Let a segmentation $\mathcal{S}$ be a partition of a set of pixels, $\mathcal{P}$, into K disjoint sets $s_1, s_2, \ldots s_K$ called segments:

$$\mathcal{S} = \{s_1, s_2, \ldots s_K\} \qquad st. \quad s_k \cap s_l = \emptyset \quad and \quad \bigcup_{k=1}^{K} s_k = \mathcal{P} \qquad \text{(C.1)}$$

Furthermore, let $N = \sum_{n=1} p_n$ and $M = \sum_{m=1} p_m$ be the sum of pixels in a segment and an image respectively. The total number of segments is usually different in the two segmentations we wish to compare, $K \neq K'$, and we assume that there are no empty clusters, $N > 0$. The confusion matrix is a $K \times K'$ table where the element (k, l) is the number of pixels in the intersection of segments $s_k \in \mathcal{S}$ and $s_l \in \mathcal{S}'$. The confusion matrix for the simple case of a pair of binary segmentations A and B for a 9 pixel image (Table C.1a-c) can be seen in Table C.1e-f.

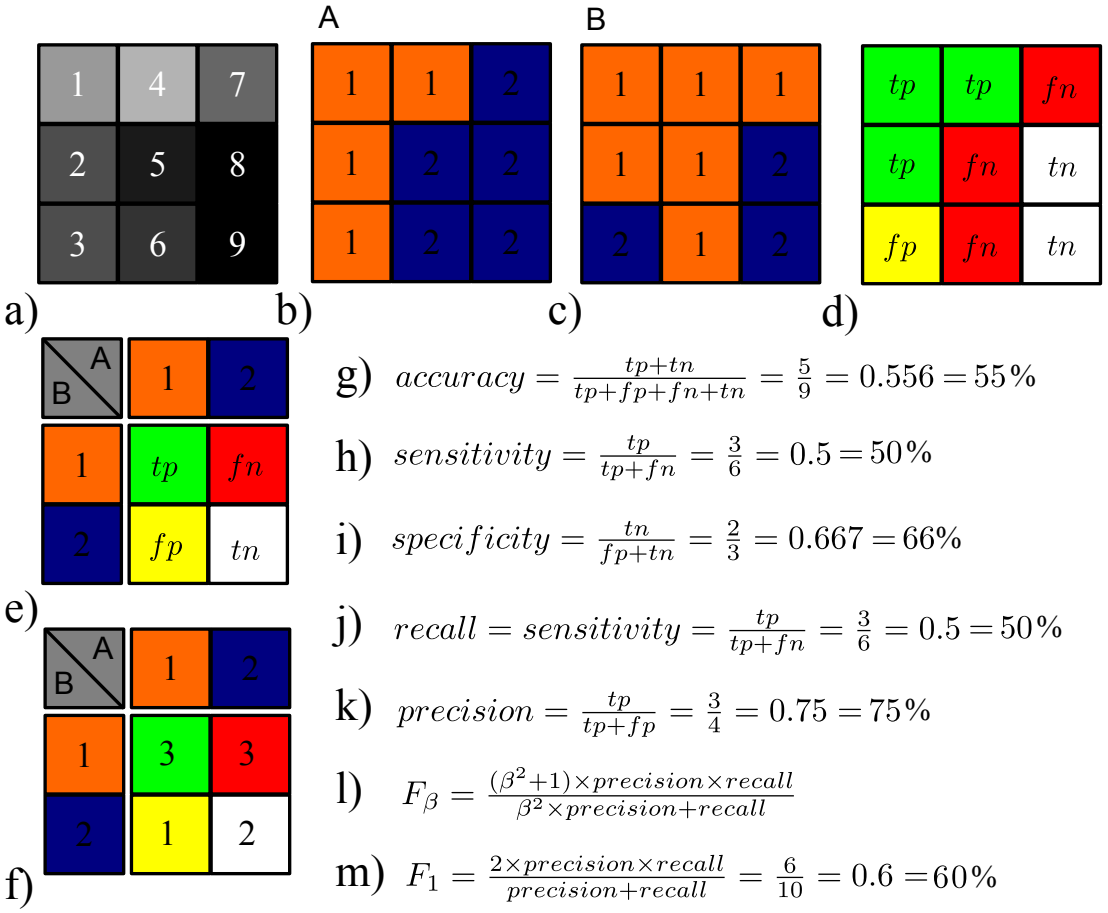


Figure C.1: Binary confusion matrix and common single pixel performance measures. a) Example 9-pixel image. b) Binary segmentation A. c) Binary segmentation B. d) Region intersections of A and B. Here B is assumed to be the correct segmentation and A that produced by an algorithm. e) Confusion Matrix. f) Confusion matrix with pixel counts for intersections shown in d). g)-m) Common single pixel performance measures.

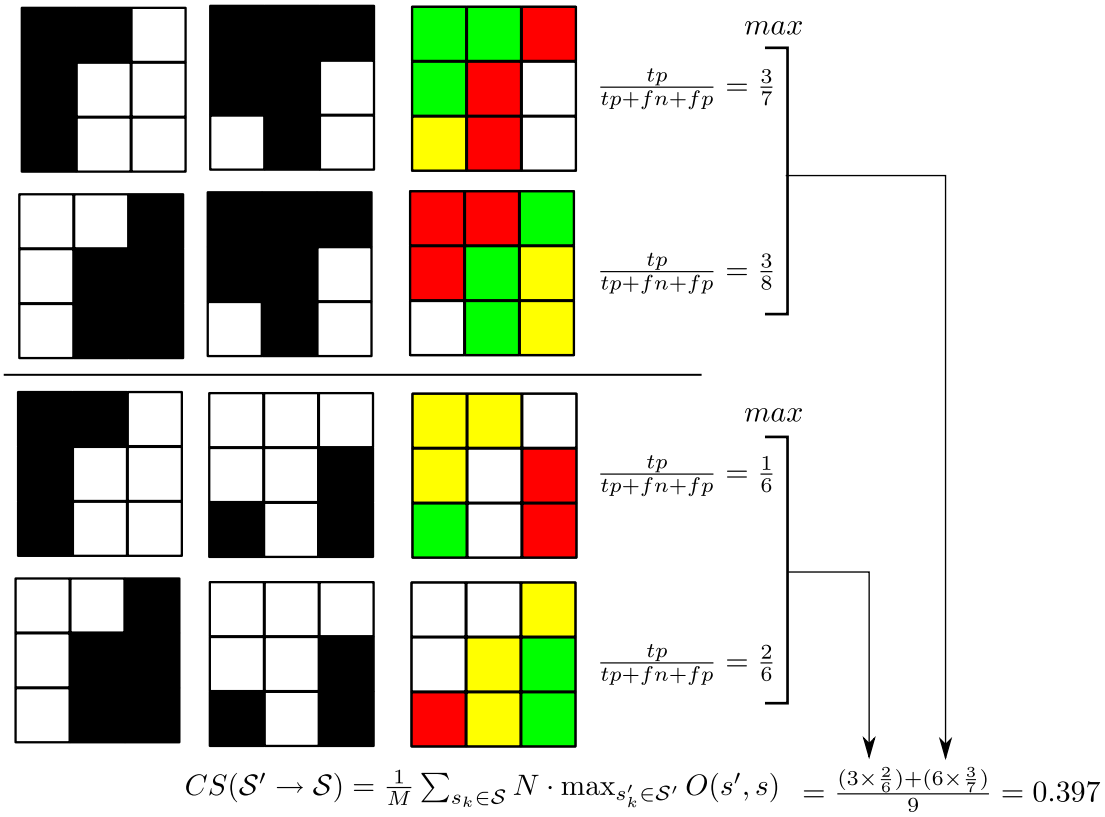


Figure C.2: Cover Score. Example of CS measure using example segmentations from Figure C.1.

Common pixel count measures that are used in the literature can be seen in Figure C.1g-m. They involve different counts of region intersections normalized into percentage errors. We will use 'Accuracy' as one measure of performance which simply counts the number of correct pixels between two segmentations (see Figure C.1g). However, for the binary forced choice comparison this measure can prove insensitive because it is usually possible to obtain large true negative counts. Results in Chapter 3 demonstrate that it is sometimes insensitive to segmentation performance.

Several different region counts are common in the literature [19], and we exploit five additional measures in our evaluation. The trade off between different measures of performance is still and ongoing area of study and useful comparisons of competing evaluation measures can be found in Sokolova et al. [244]. The problem of evaluating algorithms over multiple datasets is also examined in Demsar [69]. We review those measures that are used for evaluating results in the thesis in the following subsections.

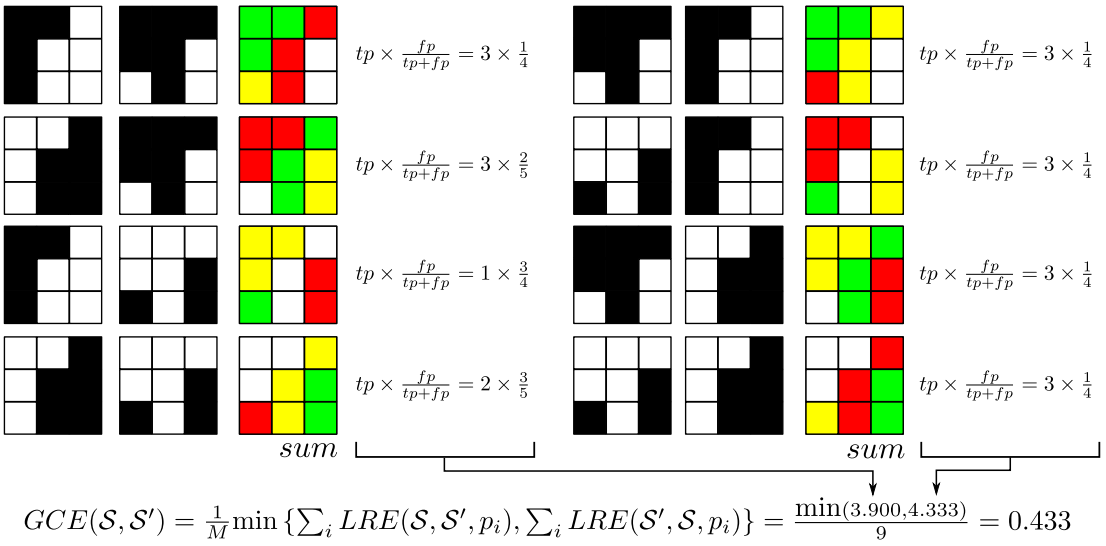


Figure C.3: Global Consistency Error. Example of GCE measure using example segmentations from Figure C.1.

C.1.1 Cover Score

One measure used to compare a segmentation $\mathcal{S}$ to the ground truth segmentation $\mathcal{S}'$ is to use a "segmentation covering" metric (CS) [19]. The overlap $O(S, S')$ between a segment in $\mathcal{S}$ and a segment in the ground truth $\mathcal{S}'$ is defined as

$$O(s, s') = \frac{|s \cap s'|}{|s \cup s'|}. \tag{C.2}$$

which returns a number that is zero if the regions do not overlap at all and one if the regions exactly match and is the same intersection/union measure used in pixel-wise recognition tasks [172, 87]. The covering of a segmentation $\mathcal{S}$ by $\mathcal{S}'$ is then given by

$$CS(\mathcal{S}' \to \mathcal{S}) = \frac{1}{M} \sum_{s_k \in \mathcal{S}} N \cdot \max_{s'_k \in \mathcal{S}'} O(s', s) \tag{C.3}$$

where K denotes the total number of regions. In other words, we find the best overlapping segment s' in the ground truth for each segment s in our segmentation, and calculate the overlap score. The segmentation cover is the average of all of these overlap scores. An example of calculating this measure for a 9 pixel example is shown in Figure C.2.

C.1.2 Global Consistency Error

Global Consistency Error (GCE) measures the extent to which regions in one segmentation S are subsets of regions in a second segmentation S'. This is commonly referred to as *refinement*. For a given pixel p_i we take the segments that contain p_i in $\mathcal{S}$ and $\mathcal{S}'$. We denote these segments $s(\mathcal{S}, p_i)$ and $s(\mathcal{S}', p_i)$ respectively. Following [173] the *local refinement error* (LRE) is then defined at a pixel p_i as:

$$LRE(\mathcal{S}, \mathcal{S}', p_i) = \frac{|s(\mathcal{S}, p_i) \backslash s(\mathcal{S}', p_i)|}{N} \tag{C.4}$$

where denotes the set differencing operator. LRE is not a symmetric and encodes *refinement* in one direction. [173] presents different ways of combining LRE at each pixel to give a measure for a whole image. In this thesis we use Global Consistency Error (GCE) that guarantees all local refinements to be in the same direction:

$$GCE(\mathcal{S}, \mathcal{S}') = \frac{1}{M} \min \left\{ \sum_i LRE(\mathcal{S}, \mathcal{S}', p_i), LRE(\mathcal{S}', \mathcal{S}, p_i) \right\} \tag{C.5}$$

Unlike our previous measures this takes a minimum value of 0 when there is no error and a maximum value of 1 when there is greatest deviation between two segmentations. An example of calculating this measure for a 9 pixel example is shown in Figure C.3.

C.1.3 Rand Index

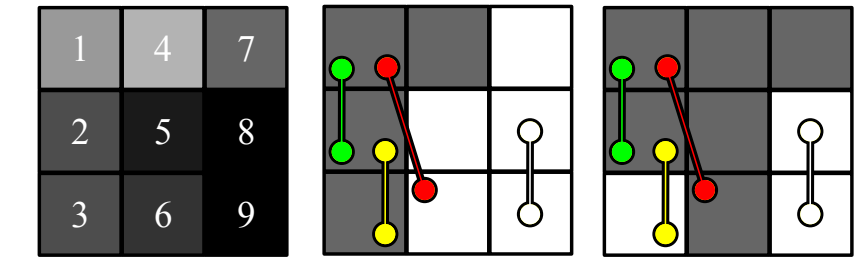
The rand index (RI), introduced by [216], measures the consistency of two segmentation by the ratio of pairs of pixels having the same label:

$$RI(\mathcal{S}, \mathcal{S}') = \frac{1}{\binom{N}{2}} \sum_{i,j,i \neq j} \left[\mathbb{I}(s_i = s_j \wedge s_i^{'} = s_j^{'}) + \mathbb{I}(s_i \neq s_j \wedge s_i^{'} \neq s_j^{'}) \right] \tag{C.6}$$

where $\mathbb{I}$ is the identity function and the denominator is the number of possible unique pairs among N data points. Note that the number of unique labels in S and S' need not be equal. The RI measure can be calculated efficiently using the region counts based on the agreement of pixel pairs, see Figure C.4:

N_{11} number of point pairs that are in the same segment under both $\mathcal{S}$ and $\mathcal{S}'$.

N_{00} number of point pairs in different segments under both $\mathcal{S}$ and $\mathcal{S}'$.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
i	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	5	5	5	5	6	6	6	7	7	8
j	2	3	4	5	6	7	8	9	3	4	5	6	7	8	9	4	5	6	7	8	9	5	6	7	8	9	6	7	8	9	7	8	9	8	9	9
N	1	1	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
	1	0	1	1	1	1	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	0	0	1	0	0	0	0	1

Figure C.4: Rand Index. Example of Rand Index [216] measure using example segmentations from Figure C.1.

N_{10} number of point pairs in the same cluster under $\mathcal{S}$ but not $\mathcal{S}'$.

N_{01} number of point pairs in the same cluster under $\mathcal{S}'$ but not $\mathcal{S}$.

These can be calculated efficiently using the confusion matrix. For example $2N_{11} = \sum_{k,k'} n_{kk'}^2 - n$. Further details of efficient calculation can be found in [177, 95].

Where multiple segmentations are available the measure can be extended to the Probabilistic Rand Index [264] by averaging over possible segmentations. We do this for the evaluations on the Berkeley Segmentation Dataset. One thing to note is that the RI measure over-estimates performance of large regions. This is often wrong for computer vision applications where incorrect estimates of small regions of the image can dramatically change the interpretation of a scene. An example of this can be see in Figure 2.13b.

C.1.4 Variation of Information

Meila [177] introduced the Variation of Information metric (VI) for the purpose of comparing general clustering algorithms. It uses the average condition entropy:

$$VI(S, S') = H(S) + H(S') - 2I(S, S') \tag{C.7}$$

where $H(S)$ is the entropy of a segmentation and $I(\mathcal{S}, \mathcal{S}')$ is the mutual information between two segmentations $\mathcal{S}$ and S':

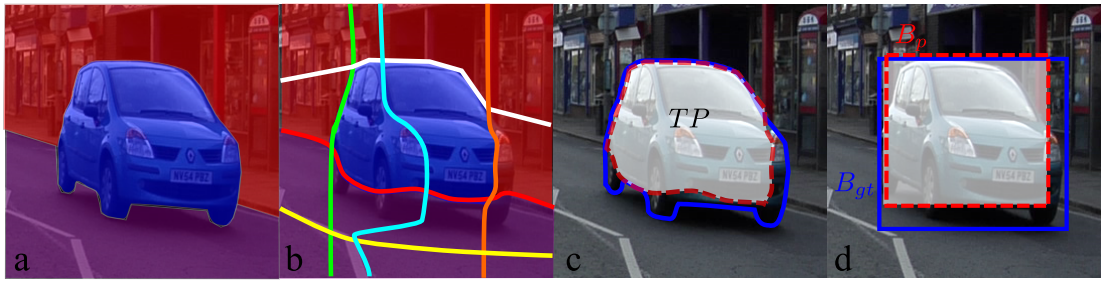


Figure C.5: Detection with over-segmentation. a) Ground truth data b) Segmentation with each superpixel labeled with mode class of ground truth data. c) Combined superpixels with true positive region highlighted. d) Bounding box overlap between ground truth and detected region.

$$H(\mathcal{S}) = -\sum_{k=1}^{K} \frac{p_k}{n} log \frac{p_k}{p} \tag{C.8}$$

$$I(\mathcal{S}, \mathcal{S}') = \sum_{k=1}^{K} \sum_{k'=1}^{K'} \frac{p_{k,k'}}{p} log \frac{p_{k,k'}}{p} \frac{p_k}{p} \frac{p_{k'}^{'}}{p} \tag{C.9}$$

$$\tag{C.10}$$

for $\mathcal{P}$ image pixels and the p_k pixels belonging to region K. The maximum value is specified by the total number of regions K in a particular segmentation, $VI \in (0, 2logK]$. Note that for small K this means that no two clusterings can be too far apart. A lower score is a better match between the two segmentations. This metric was put forward because of some of its useful theoretical properties. The perceptual meaning of this measure and its applicability to image region analysis is currently unclear [19].

C.1.5 Combined Segmentation and Detection - F_{sd}

Lastly, we introduce a measure based on combining both segmentation and detection [183]. As a measure of segmentation quality we calculate the harmonic mean of precision and recall:

$$F_s = P_s R_s / (\alpha R_s + (1 - \alpha) P_s) \tag{C.11}$$

where α is set to 0.5. We calculate object detections from our superpixel segmentation by merging together neighboring pixels where the modal ground truth class is the

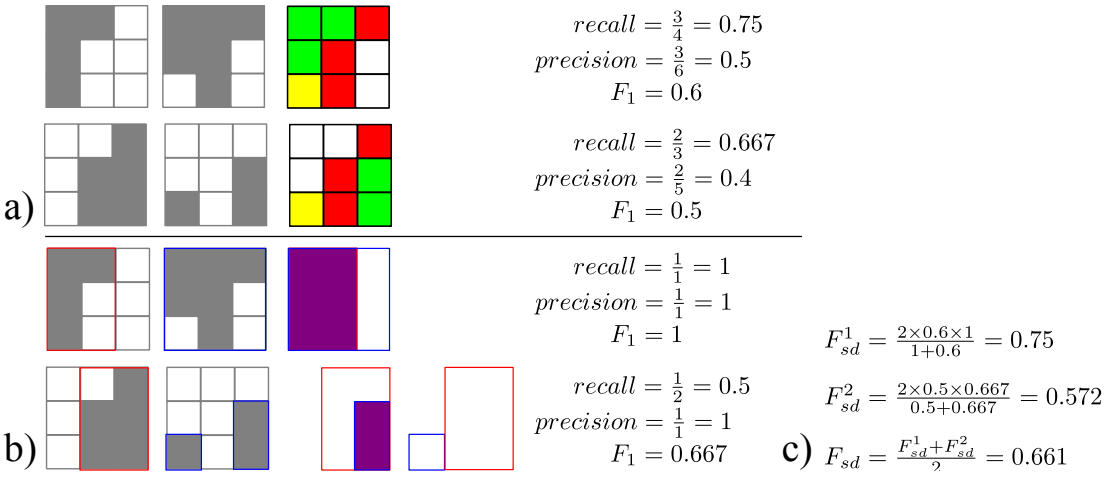


Figure C.6: F_{sd}. Example of F_{sd} measure using example segmentations from Figure C.1. a) Segmentation performance. b) Detection performance. Bounding boxes for separate regions are shown in red and blue. c) Combined segmentation and detection measures.

same (see Figure C.5). We greedily associate segmented objects to the ground truth objects and define a correct detection (hit) when the bounding boxes of the segmented and ground truth objects overlap by more than 50%. Misses constitute ground truth objects with no sufficiently overlapping segmented object. False detections constitute segmented regions that do not correspond or sufficiently overlap to ground truth regions. The overall quality of detection F_d is calculated by taking the harmonic mean of precision and recall:

$$F_d = P_d R_d / (\alpha R_d + (1-\alpha) P_d) \tag{C.12}$$

where α is set to 0.5. As neither segmentation nor detection alone completely captures our ability to parse a scene (see Figure 2.13) we treat both segmentation and detection as independent measures of performance, and use the harmonic mean of the two separate *F-measures* as our summary statistic:

$$F_{sd} = \frac{2F_s F_d}{(F_s + F_d)} \tag{C.13}$$

Equation C.13 will tend to 1 when the scene is parsed exactly. It takes into account the accuracy of the segmentation while penalizing those that split, merge or miss objects in the ground truth data. A simple 9 pixel example can be seen in Figure C.6.

One thing to note is that unlike the BSD and VOC datasets different object instances are not labeled separately in the CamVid database. This means that *detection*, and therefore F_{sd}, is overestimated on the CamVid data. However there are enough examples where object instances are separated in the ground truth data to make *detection* a useful metric of performance. An example of this is shown in Figure 4.18. It can be considered as detecting contiguous regions of the same object class.

Bibliography

[1] 2d3 Ltd.

[2] Cognex Ltd.

[3] Hawk-Eye Innovations Ltd.

[4] Kitware Inc.

[5] ObjectVideo Inc.

[6] eyeApps LLC.

[7] VectorMagic - eyeApps LLC.

[8] VitalImages Inc.

[9] Xedar Inc.

[10] Canon inc., power shot s5is advanced camera users guide. CDI-E275-010, 2007.

[11] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels. Technical report, Ecole Polytechnique Federale de Lausanne, 2010.

[12] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. *SIGGRAPH*, 2004.

[13] J. Aghajanian and S. J. D. Prince. Mosaicfaces: a discrete representation for face recognition. *WACV*, 2008.

[14] Jania Aghajanian, Jonathan Warrell, Simon J. D. Prince, Peng Li, Jennifer L. Rohn, and Buzz Baum. Patch-based within-object classification. *ICCV*, 2009.

[15] Narendra Ahuja and Sinisa Todorovic. Connected segmentation tree: a joint representation of region layout and hierarchy. *CVPR*, 2008.

[16] K. Alahari, P. Kohli, and P. H. S. Torr. Reduce, reuse & recycle: Efficiently solving mulit-label mrfs. *CVPR*, 2008.

[17] Karteek Alahari, Pushmeet Kohli, and Philip H. S. Torr. Dynamic hybrid algorithms for map inference in discrete mrfs. *TPAMI*, 32(16):1846–1857, 2010.

[18] P. Arbelaez. Boundary extraction in natural image using ultrametric contour maps. *POCV*, 2006.

[19] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contours to regions: An empirical evaluation. *CVPR*, 2009.

[20] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizeing. *ACM SIGGRAPH*, 26(3), 2007.

[21] C. B. Barber, J. P. Dobkin, and H. T. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22:469–483, 1996.

[22] Kobus Barnard, Pinar Duygulu, Raghavendra Guru, Parasad Gabbur, and David Forsyth. The effect of segmentation and feature choice in a translation model of object recognition. *CVPR*, 2003.

[23] H. Barrow and J. Tenenbaum. Recovering intrinsic scene characteristics from images. *Comp. Vision Systems*, 1978.

[24] D. Bartholomew and M. Knott. *Latent variable models and Factor analysis*. London, 1999.

[25] Dhruv Batra, Rahul Sukthankar, and Tsuhan Chen. Learning class-specific affinites for image labelling. *CVPR*, 2008.

[26] V. Belhumeur, J. Hespanha, and D. Kriegman. Eignefaces vs. fisherfaces: Recognition using class specific linear projection. *PAMI*, 19:711–720, 1998.

[27] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[28] M. Black, G. Sapiro, D. Marimont, and D. Heeger. Robust anisotropic diffusion. *IEEE Transactions on Image Processing*, 7(3):421–432, 1998.

[29] A. Blake, C. Rother, M. Perez, and Philip H. S. Torr. Interactive image segmentation using and adaptive gmmrf model. *ECCV*, LNCS 3021:428–441, 2004.

[30] Andrew Blake and Michael Isard. *Active Contours*. Springer, 1997.

[31] Eran Borenstein and Jitendra Malik. Shape guided object segmentation. *CVPR*, 1:969–976, 2006.

[32] Eran Borenstein and Shimon Ullman. Learning to segment. *ECCV*, 1:315–328, 2004.

[33] Eran Borenstein, Eitan Sharon, and Shimon Ullman. Combining top-down and bottom-up segmentation. *CVPR Workshop*, 2004.

[34] Y. Boykov and G. Funka Lea. Graph cuts and efficient n-d image segmenation. *IJCV*, 70:109–131, 2006.

[35] Yuri Boykov and Marie-Pierre Jolly. Interactive Graph Cuts for Optimal Boundry & Region Segementation of Objects in N-D Images. *ICCV*, 1:105–112, 2001.

[36] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in computer vision. *Proc. Int'l Workshop Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 359–374, 2001.

[37] Yuri Boykov and Vladamir Kolomogorov. Computing geodesics and minimal surfaces via graph cuts. *ICCV*, 2003.

[38] Yuri Boykov, Olga Veksler, and Roman Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11):1222–1239, 2001.

[39] G. Brostow, J. Fauqueur, and R. Cipolla. Semantic obje classes in video: A high-definition ground truth databse. *Pattern Recognition Letters*, 2007.

[40] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. *ECCV*, pages 44–57, 2008.

[41] A. Buades, B. Coll, and J. M. Morel. A non-local algorithm for image denoising. *CVPR*, 2:60–65, 2005.

[42] M. C. Buri, M. Weber, and P. Perona. A probabilistic approach to object recognition using local photometry and global geometry. *ECCV*, pages 628–641, 1998.

[43] John Canny. A computational approach to edge detection. *PAMI*, 8(6):679–698, 1986.

[44] Peter Carr and Richard Hartley. Minimizing energy functions on 4-connected lattices using elimination. *ICCV*, 2009.

[45] M. A. Carreira-Perpinan. Gaussian mean shift is an em algorithm. *TPAMI*, 29 (5):767–776, 2007.

[46] Miguel A Carreira-Perpinan. Fast nonparametric clustering with gaussian blurring mean-shift. *International Conference on Machine Learning*, 2006.

[47] Miguel A Carreira-Perpinan. Acceleration strategies for gaussian mean-shift image segmentation. *CVPR*, 2006.

[48] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobword: Image segmenation using expectation-maximization and its application to image querying. *Proc. Third Int. Conf. on Visual Information Systems*, 1999.

[49] V. Caselles, F. Catter, T. Coll, and F. Dibos. A geometric model for active contours in image processing. *Num Mathematik*, 66:1–31, 1993.

[50] Y. Cheng. Mean shift, mode seeking and clustering. *TPAMI*, 17:790–799, 1995.

[51] Hyoungmin Choi, Kyoung-Mu Lee, and Sang-Uk Lee. Drf-based object detection using the object adaptive pathc in the satellite imagery. *IWATIT*, 2009.

[52] Christopher M. Christoudias, Bogdan Georgescu, and Peter Meer. Synergism in lovw level cision. *Tans. 16th International Conference on Pattern Recognition*, 4:150–155, 2002.

[53] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *TPAMI*, 24(5), 2002.

[54] Thomas H. Cormen, Charles E. Lieiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. The MIT Press, 2001.

[55] Camille Couprie, Leo Grady, Laurent Najman, and Hugues Talbot ". Power watersheds: A new image segmentation framework extending graph cuts, random walker and optimal spanning forest. *ICCV*, pages 731–738, 2009.

[56] Camille Couprie, Leo Grady, Laurent Najman, and Hugues Talbot. Power watershed: A unifying graph-based optimization framework. *PAMI*, 2011 - preprint.

[57] Michel Couprie and Gilles Bertrand. Topological grayscale watershed transformation. *IN SPIE VISION GEOMETRY V PROCEEDINGS*, 3168:136–146, 1997.

[58] Timothee Cour and Jianbo Shi. Recognizing objects by piecing together ther segmentation puzzel. *CVPR*, 2007.

[59] Timothee Cour, Florence Benezit, and Jianbo Shi. Spectral segmentation with mulitscale graph decomposition. *CVPR*, 2:1124–1131, 2005.

[60] J. Cousty, G. Bertrand, L. Najman, and M. Couprie. Watershed cuts: Minimum spanning forests and the drop of warter principle. *PAMI*, 31:1362–1374, 2009.

[61] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *CVPR*, 2005.

[62] E. R. Davies. Design of cost-effective systems for the inspection of certain food products dugin manufacture. *Proc. 4th Int. Conf. on Robot Vision and Sensory Control*, pages 437–446, 1984.

[63] Jean-Charles de Borda. Memoire sur les election au scrutin. *Historie de l'Academie Royal Sciences*, 1781.

[64] B. Delaunay. Sur la sphre vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, 7:793–800, 1934.

[65] Andew Delong, Anton Osokin, Hossam N. Isack, and Yuri Boykov. Fast approximate energy minimization with label costs. *CVPR*, 2010.

[66] Andrew Delong and Yuri Boykov. A scaleable graph-cut algorithm for n-d grids. *CVPR*, 2008.

[67] Andrew Delong and Yuri Boykov. Globally optimal segmentation of multi-region objects. *ICCV*, 2009.

[68] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39 (1):1–38, 1977.

[69] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[70] Yining Deng and B. S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *TPAMI*, 23:800–810, 2001.

[71] H. Digabel and C. Lantuejoul. Iterative algorithms. *European Symp. Quantitative Analysis of Microstructures in Material Science, Biology and Medicine*, pages 85–89, 1978.

[72] Edgar W. Dijkstra. A note on two problems in connexion of graphs. *Numerishe Mathematik*, 1:269–271, 1959.

[73] E. A. Dinic. Algorithm for solutoin of a problem of maximum flow in networks with power estimation. *Soviet Math. Deokl*, 11:1277–1280, 1970.

[74] Piotr Dollr, Zhuowen Tu, and Serge Belongie. Supervised learning of edges and object boundaries. *CVPR*, 2:1964–1971, 2006.

[75] Justin Domke, Alap Karapurkar, and Yaiannis Aloimonos. Who killed the directed model. *CVPR*, 2008.

[76] P. Drineas, A. Frieze, E. Kannan, S. Vempalal, and V. Vinay. Clustering large graphs via the singular value decomposition. *Machine Learning*, 56:9–33, 1999.

[77] Fabio Drucker and John MacCormick. Fast superpixels for video analysis. *Proceedings of IEEE Workshop on Motion and Video Computing*, 2009.

[78] Richard O. Duda, Pete E. Hart, and David G. Stork. *Pattern Classification*, volume 24. New York: John Wiley & Sons, 2001. doi: http://dx.doi.org/10.1007/s00357-007-0015-9. ISBN: 0-471-05669-3.

[79] Cynthia Dwork, Ravi Kumar, Moni Naor, and D Sivakumar. Rank aggregation methods for the web. *Proceedings of the 10th international conference on World Wide Web*, pages 613–622, 2001.

[80] A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. *SIGGRAPH*, pages 341–346, 2001.

[81] Alexei A Efros and Thomas K Leung. Texture synthesis by non-paramteric sampling. *ICCV*, 2:1033–1038, 1999.

[82] M. Etoh and Y. Shirai. Segmentation and 2d motion estimation by reigon fragments. *ICCV*, pages 98–104, 1993.

[83] M. Everingham, H. Muller, and B.T. Thomas. Algorithm evaluation by probabilistic fitness/cost analysis and application to image segmentation. *ACCV2002*, pages 580–586, 2002.

[84] Mark Everingham, Henk Muller, and Barry T. Thomas. Evaluating image segmentation algorithms using monotonic hulls in fitness/cost space. *BMVC*, pages 363–372, 2001.

[85] Mark Everingham, Henk Muller, and Barry T. Thomas. Evaluation of image segmenation algorithms using the monotonic hulls. Technical report, University of Bristol, 2001.

[86] Mark Everingham, Henk Muller, and Barry T. Thomas. Evaluating image segmentation algorithms using the pareto front. *ECCV*, IV:34–48, 2002.

[87] Mark Everingham, Luc Van Gool, Christopher K I Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes VOC challenge. *IJCV*, 2009.

[88] Pedro Felzenszwalb and Daniel Huttenlocher. Efficient Graph-Based Image Segmentation. *IJCV*, 2(59):167–181, 2004. URL `http://www.cs.cornell.edu/ dph/`.

[89] Pedro Felzenszwalb and Daniel Huttenlocher. Pictorial structures for object recognition. *IJCV*, 2005.

[90] Pedro Felzenszwalb and David McAllester. A min-cover approach for finding salient curves. *Proceedings of the 7th IEEE Computer Society Workshop on Perceptual Organization in Computer Vision (POCV)*, 2006.

[91] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient belief propagation for early vision. *CVPR*, Vol.1:261–268, 2004.

[92] Pedro F. Felzenszwalb and Olga Veksler. Tiered scene labeling with dynamic programming. *CVPR*, 2010.

[93] Pedro F. Felzenszwalb and Ramin Zabih. Dynamic programming and graph algorithms in computer vision. *to appear PAMI*, 2010.

[94] Robert Fergus, P. Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. *ECCV*, LNCS 3021:242–256, 2004.

[95] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierachical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983.

[96] J. Freixenet, X. Munoz, D. Raba, J. Marti, and X. Cuff. Yet another survey on image segmentation: Region and boundary information integration. *ECCV*, pages 408–422, 2002.

[97] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. *ICCV*, 2009.

[98] K. Funkunaga and L. D. Hosteler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transaction on Information Theory*, 21:32–40, 1975.

[99] Carolina Galleguillos, Boris Babenko, Andrew Rabinovich, and Serge Belongie. Weakly supervised object localization with stable segmentations. *ECCV*, 2008.

[100] Carolina Galleguillos, Andrew Rabinovich, and Serge Belongie. Object categorization using co-occurrence, location and appearance. *CVPR*, 2008.

[101] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and bayesian restoration of images. *PAMI*, 6:721–741, 1984.

[102] Carol M Ginsberg and Delle Maxwell. Graphical marionette. *ACM SIGGRAPH*, 1:172–179, 1983.

[103] Stas Goferman and Lihi Zelnik-Manor. Context-aware saliency detection. *CVPR*, 2010.

[104] G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins Press, 1989.

[105] Stephen Gould, Jim Rodgers, David Cohen, Gal Elidan, and Daphne Koller. Multi-class segmentation with relative location prior. *IJCV*, 2008.

[106] J Gower and G Ross. Minimum spanning trees and single linkage clustering. *Applied Statistics*, 18(1):54–64, 1969.

[107] Leo Grady. Mulitlabel random walker image segmentation using prior models. *CVPR*, 1:763–770, 2005.

[108] Leo Grady. Random walks for image segmentation. *TPAMI*, 28(11), 2006.

[109] Leo Grady and Ali K. Sinop. Fast approximate random walker segmeantation using eigenvector precomputation. *CVPR*, 2008.

[110] Leo John Grady. *Space-variant computer vision: a graph theoretic approach*. PhD thesis, Boston University, 2004.

[111] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, 51(2):271–279, 1989.

[112] Yanlin Guo, Cen Rao, Supun Smaraskera, Janet Kim, Rakesh Kumar, and Harpreet Sawhney. Matching vehicles under large post transofrmations using approximate 3d models and piecewise mrf model. *CVPR*, 2008.

[113] J. M. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. *Unpublished manuscript.*, 1971.

[114] Assaf Harel, Shimon Ullman, Boris Epshtien, and Shlomo Bentin. Mutal information of image fragments predicts categorization in humans: Electrophysiological and behavioral evidnence. *Vision Research*, 47:2010–2020, 2007.

[115] Xuming He, Richard D. Zemel, and Miguel A. Carreira-Perpinan. Multiscale Conditional Random Fields for Image Labeling. *CVPR*, 2:695–702, 2004.

[116] Xuming He, Richard S. Zemel, and Debajyoti Ray. Learning and incorporating top-down cues in image segmentation. *ECCV*, 1:338–351, 2006.

[117] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 2002.

[118] D. Hoiem, A.A. Efros, and M. Hebert. Automatic photo pop-up. *ACM SIGGRAPH*, 2005.

[119] D. Hoiem, A.A. Efros, and M. Hebert. Putting objects in perspective. *CVPR*, 2006.

[120] D. Hoiem, A.A. Efros, and M. Hebert. Closing the loop on scene interpretation. *CVPR*, 2008.

[121] Derek Hoiem, Alexi A Efros, and Martial Hebert. Geometric context from a single image. *ICCV*, pages 654–661, 2005.

[122] Derek Hoiem, Alexiei A. Efros, and Martial Herbert. Recovering surface layout from an image. *IJCV*, 75(1):151–172, 2007.

[123] Derek Hoiem, Carsten Rother, and John Winn. 3D layout CRF for multi-view object class recognition and segmentation. *CVPR*, 2007.

[124] Derek Hoiem, Andrew N. Stein, Alex A. Efros, and Martial Hebert. Recovering occlusion boundaries from a single image. *ICCV*, 1:1–8, 2007.

[125] P. W. Holland and R. E. Welch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics: Theory and Methods*, A6:813–827, 1977.

[126] Rui Huang, Vladimir Pavlovic, and Dimitris N. Metaxas. A graphical model framework for coupling mrfs and deformable models. *CVPR*, 2:739–746, 2004.

[127] Peter J. Huber. *Robust Statistical Procedures*. SIAM, 1977.

[128] Hiroshi Ishikawa. Exact optimization for markov random fields with convex priors. *TPAMI*, 25(10):1333–1336, 2003.

[129] Hiroshi Ishikawa. Transformation of general binary mrf minimization of the first order case. *PAMI*, Preprint:–, 2010.

[130] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31:651–666, 2010.

[131] Allan Jepson and Michael Black. Mixture models for optical flow computation. *CVPR*, pages 760–761, 1993.

[132] Michael Kass, Andrew Witkin, and Demetri Terzopoulous. Snakes: Active contour models. *IJCV*, 1(4):321–331, 1988.

[133] M. Kendall. A new meaure of rank correlation. *Biometrika*, 30:81–89, 1938.

[134] B. Kimia, A. Tannenbaum, and S. Zucker. Toward a computational theory of shape: An overview. *Lecture Notes in Computer Science*, 427:402–407, 1990.

[135] Pushmeet Kohli and Philip H. S. Torr. Measuing uncertainty in graph cut solutions - efficiently computing min-marginal energies using dynamic graph cuts. *ECCV*, 2006.

[136] Pushmeet Kohli, M. Pawan Kumar, and Philip H.S. Torr. P3 and beyond: Solving energies with higher order cliques. *CVPR*, 2007.

[137] Pushmeet Kohli, M Pawan Kumar, and Philip H.S. Torr. P3 & beyond: Move making algorithms for solving higher order functions. *PAMI*, 31:1645–1656, 2008.

[138] Pushmeet Kohli, Lubor Ladicky, and Philip Torr. Robust higher order potentials for enforcing label consistency. *CVPR*, 2008.

[139] Pushmeet Kohli, Lubor Ladicky, and Philip H. S. Torr. Robust higher order potentials for enforcing label consistency. *IJCV*, 82(3):302–324, 2009.

[140] Vladamir Kolmogorov and Yuri Boykov. What metrics can be approximated by geo-cuts, or global optimization of length/area and flux. *ICCV*, 1:564–571, 2005.

[141] Vladamir Kolomogorov and Ramin Zabih. What energy functions can be minimzed via graph cuts? *PAMI*, 26:147–159, 2004.

[142] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. American Mathematical Society*, 7(1):48–50, 1956.

[143] M. Pawan Kumar, Philip H. S. Torr, and Andrew Zisserman. Obj cut. *CVPR*, 1: 18–25, 2005.

[144] Neeraj Kumar, Li Zhang, and Shree Nayar. What is a good nearest neighbour algorithm for finding similar patches in images? *ECCV*, 2:364–378, 2008.

[145] Sanjiv Kumar and Martial Hebert. A hierarchical field framework for unfied context-based classification. *ICCV*, 2005.

[146] Sanjiv Kumar, Jonas Sugust, and Martial Herbert. Exploiting inference for approximate parameter learning in discriminative fields: An empirical study. *EMMCVPR*, 2005.

[147] Vivek Kwatra, Arno Schdl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM SIGGRAPH*, 22(3): 277–286, 2003.

[148] Lubor Ladicky, Chris Russell, Pushmeet Kohli, and Philip Torr. Associative hierachical crfs for object class image segmentation. *ICCV*, 2009.

[149] Lubor Ladicky, Paul Sturgess, Karteek Alahari, Chris Russell, and Philip H. S. Torr. What, where and how many? combining object detectors and crf's. *ECCV*, 2010.

[150] Lubor Ladicky, Paul Sturgess, Chris Russell, Sunando Sengupta, Yalin Bastanlar, William Clocksin, and Philip H. S. Torr. Joint optimization for object class segmentation and dense stereo. *BMVC*, 2010.

[151] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282 – 289, 2001.

[152] X Lan, Stefan Roth, Daniel P. Huttenlocher, and Michael J. Black. Efficient belief propagation with learned higher-order markov random fields. *ECCV*, 2: 269–282, 2006.

[153] S. Lauristzen. *Graphical Models*. Oxford University Press, 1996.

[154] Eugene L. Lawler. *Combinatorial Optimization: Networks and Matroids*. 1976.

[155] Victor Lempitsky, Stefan Roth, and Carsten Rother. Fusion-flow: Discrete-continuous optimization for opitcal flow fields. *CVPR*, 2008.

[156] Victor Lemptisky, Carthen Rother, and Andrew Blake. Logcut - efficient graph cut optimization for markov random fields. *ICCV*, 2007.

[157] Anat Levin and Yair Weiss. Learning to combine bottom-up and top-down segmentation. *ECCV*, 1:581–594, 2006.

[158] M D Levine and A Nazif. Dynamic measurement of computer generated image segmentations. *PAMI*, 7:155–164, 1985.

[159] Alex Levinshtein, Adrian Stere, Kiriakos Kutulakos, David J. Fleet, Sven J. Dickinson, and Kaleem Siddiqi. Turbopixels: Fast superpixels using geometric flows. *TPAMI*, 31(12), 2009.

[160] G. Li. *Exploring Data Tables, Trends and Shapes*. Wiley, 1985.

[161] Kang Li, Xiaodong Wu, Danny Z Chen, and Milan Sonka. Optimal surface segmentation in volumetric images - a graph theoretic approach. *TPAMI*, 28(1): 119–134, 2006.

[162] L.J. Li and L. Fei-Fei. What, where and who? classifying event by scene and object recognition . *ICCV*, 2007.

[163] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *ACM SIGGRAPH*, 2004.

[164] T. Lindeberg. *Scale-Spcae Theory in Computer Vision*. Boston, MA, 1994.

[165] Jiangyu Liu and Jian Sun. Parallel graph-cuts by adaptive bottom-up merging. *CVPR*, 2010.

[166] Xiaoqing Liu, Olga Veksler, and Jagath Samarabandu. Order preserving moves for graph cut based optimization. *TPAMI*, 32, 2008.

[167] Xiaoqing Liu, Olga Veksler, and Jagath Samarabandu. Order preserving moves for graph cut based optimization. *TPAMI*, 32, 2010.

[168] S. P. Lloyd. Least sqaures quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[169] Herve Lombaert, Yiyong Sun, Leo Grady, and Chenyang Xu. A multi-level banded graph cuts method for fast image segmentation. *ICCV*, 1:1550–5499, 2005.

[170] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[171] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. *CVPR*, 2008.

[172] Tomasz Malisiewicz and Alexei A. Efros. Improving spatial support for objects via multiple segmentations. *BMVC*, 2007.

[173] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *ICCV*, 2:416–423, 2001.

[174] David R. Martin, Charless C. Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549, 2004.

[175] Jiri Matas and Joseph Kittler. Spatial and feature space clustering: Applications in image analysis. *Computer Analysis of Images and Patterns*, pages 162–173, 1995.

[176] Peter Meer and Bogdan Geogescu. Edge detection with embedded confidence. *PAMI*, 23:1351–1365, 2001.

[177] M. Meila. Comparing clusterings by the variation of information. *Proceedings of the Sixteenth Annual Conference of Computational Learning Theory (COLT). Springer.*, 2003.

[178] M. Meila. Comparing clusterings: An axiomatic view. *ICML*, 2005.

[179] Branislav Micusik and Jana Kosecka. Semantic segmentation of street scenes by superpixel co-occurrence and 3d geometry. *VOEC*, 2009.

[180] Branislav Micusik and Jana Kosecka. Multi-view superpixel stereo in urban environments. *IJCV*, 89:106–119, 2010.

[181] Umar Mohammed, Simon J. D. Prince, and Jan Kautz. Visio-lization: Generating novel facial images. *ACM SIGGRAPH*, 2009.

[182] Alastair P. Moore, Simon J. D. Prince, Jonathan Warrell, Umar Mohammed, and Graham Jones. Superpixel lattices. *CVPR*, 2008.

[183] Alastair P. Moore, Simon J. D. Prince, Jonathan Warrell, Umar Mohammed, and Graham Jones. Scene shape priors for superpixel segmentation. *ICCV*, 2009.

[184] Alastair P. Moore, Simon J. D. Prince, and Jonathan Warrell. Lattice cut - constructing superpixels using layer constraints. *CVPR*, 2010.

[185] G. Mori. Guiding model search using segmentation. *ICCV*, 2:1417–1423, 2005. URL `http://www.cs.sfu.ca/ mori/research/superpixels/`.

[186] Greg Mori, Xiaofeng Ren, Alexi A. Efros, and Jitendra Malik. Recovering human body congurations: combining segmentation and recognition. *CVPR*, 2: 326–333, 2004.

[187] Eric N. Mortensen and William A. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60(5):349–384, 1998.

[188] Vida Movahedi and James H. Elder. Design and perceptual validation of performance measures for salient object segmentation. *Proceedings of the 7th IEEE Computer Society Workshop on Perceptual Organization in Computer Vision (POCV)*, 2010.

[189] Yadon Mu, Bingfeng Zhou, and Shuicheng Yan. Information-theoretic analysis of input strokes in visual object cutout. *IEEE Transaction on Multimedia*, 2010.

[190] F. Murtagh. Complexities of hierarchic clustering algorithms: the state of the art. *Computational Statistics Quarterly*, 1:101–113, 1984.

[191] Makatot Nagao, Takashi Matasuyama, and Yoshio Ikeda. Region extraction and shape analysis in aerial photographs. *Computer Graphics and Image Processing*, 10(3):195–223, 1979.

[192] Laurent Najman and Michel Schmitt. Geodesic saliency of watershed contours and hierachical segmentation. *TPAMI*, 18:1163–1173, 1996.

[193] Ramanan Navaratnam, Andrew W. Fitzgibbon, and Roberto Cipolla. The joint manifold model for semi-supervised multi-valued regression. *ICCV*, 2007.

[194] Shree K. Nayar. Computational cameras: Redfining the image. *IEEE Computer Magazine, Special Issue on Computational Photography*, pages 30–38, 2006.

[195] Jerzy Neyman and Egon Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A*, 231:289337, 1933.

[196] Ohta, Takeo Kanade, and T. Sakai. An analysis system for scenes containing objects with substrucutres. *Prococeedings of the Fourth International Joint Conference on Pattern Recognition*, pages 752–754, 1978.

[197] Ryuzo Okada and Stefano Soatto. Relevant feature selection for human post estimation and localization in clusttered images. *ECCV*, 2008.

[198] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001.

[199] Patrick Ott and Mark Everingham. Implicit color segmentation features for pedestrian and object detection. *ICCV*, 2009.

[200] Caroline Pantofaru, Cordelia Schmid, and Martial Hebert. Object recognition by integrating multiple image segmentations. *ECCV*, pages 481–494, 2008.

[201] Sylvain Paris. Edge-preserving smoothing and mean-shift segmentation of video streams. *ECCV*, 2008.

[202] Sylvain Paris and Fredo Durand. A topological approach to hierachical segmentation using mean shift. *CVPR*, 2007.

[203] Patrick Perez and Fabrice Heitz. Restriction of a markov random field on a graph and mulitresolution statistical image modelling. *Tans. Information Theory*, 42 (1):180–190, 1996.

[204] Judea Perl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, 1988.

[205] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *TPAMI*, 12:629–639, 1990.

[206] F. Perronnin, J. L. Dugelay, and K. Rose. Iterative decoding of two dimensional hidden markov models. *ICASSP*, 3(3):329–332, 2003.

[207] Fatih Porikli. Constant time o(1) bilateral filtering. *CVPR*, 2008.

[208] Mukta Prasad, Andrew Zisserman, and Andrew Fitzgibbon. Learning class-specific edges for object detection and segmentation. *ICVGIP*, 2006.

[209] Simon J. D. Prince and Jania Aghajanian. Gender classification in uncontrolled stettings using additive logistic models. *ICIP*, 2009.

[210] Simon J. D. Prince and James H. Elder. Probabilistic linear discriminant analysis for inferences about indetity. *ICCV*, 2007.

[211] Simon J. D. Prince and James H. Elder. Baysian identitiy clustering. *Computer and Robot Vision*, pages 32–39, 2010.

[212] Andrew Rabinovich, Tilman Lange, Joachim M. Buhmann, and Serge Belongie. Model order selection and cue combination for image segmentation. *CVPR*, 2006.

[213] Andrew Rabinovich, Andrea Vedaldi, and Serge Belongie. Does image segmentation improve object catagorization? Technical report, University of California, San Diego - Technical Report CSE CS2007-0908, 2006.

[214] Andrew Rabinovich, Andrea Vedaldi, Caroline Galleguillos, Eric Wiewiora, and Serge Belongie. Objects in context. *ICCV*, 2007.

[215] Srikumar Ramalingam, Pushmeet Kohli, Karteek Alahari, and Philip H. S. Torr. Exact inference in mulit-label crfs with higher order cliques. *CVPR*, 2008.

[216] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Jornal of the American Statistical Association*, 66:846–850, 1971.

[217] A. L Ratan, O. Maron, W. E. L. Grimson, and T. Lozano-Perez. A framework for learning query concepts in image classification. *CVPR*, pages 423–431, 1999.

[218] Xianfeng Ren and Jitendra Malik. Learning a classification model for segmentation. *ICCV*, 1(1):10–17, 2003.

[219] Xianfeng Ren, Chales C. Fowlkes, and Jitendra Malik. Scale-invariant contour completion using conditional random fields. *ECCV*, 2005.

[220] Xianfeng Ren, Chales C. Fowlkes, and Jitendra Malik. Figure/ground assignment in natural images. *ECCV*, 2:614–627, 2006.

[221] Xiaofeng Ren. Multi-scale improves boundary detection in natural images. *ECCV*, 2008.

[222] C. Van Rijsbergen. *Information Retrieval*. Department of Computer Science, University of Glasgow, 1979.

[223] Noah Robischon. Smile, gamers: Youre in the picture. *The New York Times*, page G1, 2003.

[224] Stefan Roth and Michael J. Black. Fields of experts: A framework for learning image priors. *CVPR*, 2:860–867, 2005.

[225] Carsten Rother, Vladamir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph*, 23:309314, 2004.

[226] David Rowan. Kinect for Xbox 360: The inside story of Microsoft's secret 'Project Natal'. *Wired Magazine, November*, 2010. URL `http://www.wired.co.uk/magazine/archive/2010/11/features/`.

[227] Bryan C. Russell, Alexei A. Efros, Josef Sivic, William T. Freeman, and Andrew Zisserman. Using multiple segmentations for discover objects and their extent in image collections. *CVPR*, 2006.

[228] Chris Russell, Lubor Ladicky, Pushmeet Kohli, and Philip H. S. Torr. Exact and approximate inference in associative hierachical networks using graph cuts. *The 26th Conference on Uncertainty in Artificial Intelligence*, 2010.

[229] M.E. Sargin, L. Bertelli, B.S. Manjunath, and K. Rose. Probabilistic occlusion boundary detection on spatio-temporal lattices. *ICCV*, 2009.

[230] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Learning 3d scene structure from a single still image. *ICCV workshop on 3D Representation for Recognition*, 2007.

[231] D. Schlesinger and B. Flach. Transforming an arbitrary minsum problem into a binary one. *Dresden Univ. of Technology*, Technical Report TUD-FI06-01, 2006.

[232] Henry Schneiderman and Takeo Kanade. A statistical method for 3d object detection applied to faces and cars. *CVPR*, 2000.

[233] Nicol N. Schraudolph. Polynomial-time exact inference in np-hard binary mrfs via reweighted perfect matching. *International Conference on Artificial Intellignece and Statistics (AISTATS)*, 9:717–724, 2010.

[234] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[235] Eitan Sharon, Achi Brandt, and Ronen Basri. Fast mulitscale image segmentation. *CVPR*, 1:70–77, 2000.

[236] Eitan Sharon, Achi Brandt, and Ronen Basri. Segmentation and boundary detection using mulitscale intensity measurements. *CVPR*, 1:469–476, 2001.

[237] Yaser A. Sheikh, Erum A. Khan, and Takeo Kanade. Mode-seeking by medoidshifts. *ICCV*, 2007.

[238] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *TPAMI*, 22(8):888–905, 2000. URL `http://www.cs.berkeley.edu/ malik/papers/SM-ncut.pdf`.

[239] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. *ECCV*, 1:1–15, 2006.

[240] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. *CVPR*, 2008.

[241] A. K. Sinop and Leo Grady. A seeded image segmentation framework unifying graph cuts and radnom walker which yields a new algorithm. *ICCV*, 2007.

[242] Pierre Soille. Constrained connectivity for hierarchical image partitioning and simplification. *TPAMI*, 30:1132–1145, 2008.

[243] R. Sokal and C. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38:1409–1438, 1958.

[244] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc; a family of discriminant measures for performnace evaluation. *AI 2006: Advances in Artificial Intelligence*, II:1015–1021, 2006.

[245] C. Spearman. Demonstration of formulae for true measurement of correlation. *The American Journal of Phsychology*, 18(2):161–169, 1907.

[246] Chris Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. *CVPR*, 1:333–339, 1999.

[247] Petter Strandmark and Fredrik Kahl. Parallel and distributed graph cuts by dual decomposition. *CVPR*, 2010.

[248] Paul Sturgess, Karteek Alahari, Lubor Ladicky, and Philip H. S. Torr. Combining appearance and structure from motion feature for road scene understanding. *BMVC*, 2009.

[249] C. Sutton and A. McCallum. Priecewise training of undirected models. *Proc. Conf. on Uncertainty in Artificial Intellignece*, 2005.

[250] Martin Szummer, Pushmeet Kohli, and Derek Hoiem. Learning cfrs using graph cuts. *ECCV*, 2008.

[251] Mark Tabb and Narendra Ahuja. Mulitscale image segmentation by integrated edge and region detection. *Transactions on Image Processing*, 6:642–655, 1997.

[252] Hai Tao, Harpreet S. Sawhney, and Rakesh Kumar. A global matching framework for stereo computation. *ICCV*, 2001.

[253] Stanford Racing Team. Stanford's robotic vehicle "junior": Interim report. Technical report, Standford University, 2007.

[254] J. M. Tenenbaum and H. Barrow. Experiments in interpretation guided segmentation. *Artificial Intellignece*, 8:241–274, 1977.

[255] Mikkel Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM*, 46(3):362–394, 1999.

[256] Tijmen Tieleman and Geoffrey Hinton. Using fast weights to improve persistent constrastive divergence. *Proc. International Conference on Machine Learning*, 2009.

[257] Sinisa Todorovic and Narendra Ahuja. Extracting subimages of an unknown catagory from a set of images. *CVPR*, 1:927–934, 2006.

[258] David A. Tolliver and Gary L. Miller. Graph partitioning by spectral rounding: Applications in image segmentation and clustering. *CVPR*, 1:1053–1060, 2006.

[259] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. *ICCV*, pages 836–846, 1998.

[260] Antonio Torralba. Contextual priming for object detection. *IJCV*, 53(2):169–191, 2003.

[261] Z. Tu. Probabilistic boosting-tree: Learning discriminative models for classification, recognition and clustering. *ICCV*, 2005.

[262] M. Turk and A. Pentland. Face recognition using eigenfaces. *CVPR*, pages 586–591, 1991.

[263] Shimon Ullman, Michel Vidal-Naquet, and Erez Sali. Visual features of intermediate complexity and their use in classification. *Nature: Neuroscience*, 5(7): 682–687, 2002.

[264] Ranjith Unnikrishnan, Caroline Pantofaru, and Martial Hebert. Toward objective evaluation of image segmentation algorithms. *TPAMI*, 29(6):929–944, 2007.

[265] R. Urquhart. Graph theoretical clustering based on limited neighbour sets. *Pattern Recognition*, 15(3):173–187, 1982.

[266] Anton van den Hengel, Anthony Dick, Thorsten Thormahlen, Ben Ward, and Philip H. S. Torr. Videotrace: Rapid interactive scene modelling from video. *ACM SIGGRAPH*, 26(86), 2007.

[267] Andrea Vedaldi and Steffano Soatto. Quick shift and kernel methods for mode seeking. *ECCV*, 2008.

[268] Olga Veksler, Yuri Boykov, and Paria Mehrani. Superpixels and supervoxels in an energy optimization framework. *ECCV*, 2010.

[269] Luc Vincent and Pierre Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *PAMI*, 13(6):583–598, 1991.

[270] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 1:511–518, 2001.

[271] R. Wallace, P. W. Ong, and E. Schwartz. Space variant image processing. *IJCV*, 13(1):71–90, 1994.

[272] Jue Wang, Pravin Bhat, R. Alex Colburn, Maneesh Agrawala, and Michael F. Cohen. Interactive video cutout. *ACM SIGGRAPH*, 1:585–594, 2005.

[273] Wei Wang and Yizhou Wang. Measuing visual saliency by site entropy rate. *CVP*, 2010.

[274] Jonathan Warrell, Alastair P. Moore, and Simone J. D. Prince. Vistas: Hierarchical boundary priors using multiscale conditional random fields. *BMVC*, 2009.

[275] Jonathan Warrell, Simon J. D. Prince, and A. P. Moore. Epitomized priors for mulit-labeling problems. *CVPR*, 2009.

[276] Jonathan Warrell, Simon Prince, and Philip Torr. Styp-boost: A bilinear boosting algorithm for learning style-parameterized classifiers. *BMVC*, 2010.

[277] John Watkinson. *The MPEG Handbook*. Elsevier Ltd., 2006.

[278] M Weber, Max Welling, and P. Perona. Unsupervised learning of models for recognition. *ECCV*, pages 18–32, 2000.

[279] Yair Weiss and Edward H. Adelson. A unified mixture framework for motion segmentation: incorporating spatial coherence and estimating the number of model. *CVPR*, pages 321–326, 1996.

[280] Thomas Windheuser, Thomas Schoenemann, and Daniel Cremers. Beyond connecting the dots: A polynomial-time algorithm for segmentation and boundary estimation with imprecise user input. *ICCV*, 2009.

[281] John Winn. Probabilistic models for understanding images. *ICML*, Presentation, 2008.

[282] John Winn and N Jojic. Locus: Learning object classes with unsupervised segmentation. *ICCV*, 1:753–763, 2005.

[283] John Winn and Jamie Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. *CVPR*, 1:37–44, 2006.

[284] Oliver Woodford, Philip Torr, Ian Reid, and Andrew Fitzgibbon. Global stereo reconstruction under second order smoothness priors. *PAMI*, 24(6), 2009.

[285] Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *TPMAI*, 15(11): 1101–1113, 1993.

[286] Changobo Yang, Ming Dong, and Jing Hua. Region-based image annotation using asymmetrical support vector machine-based multiple-instance learning. *CVPR*, 2006.

[287] Lin Yang, Peter Meer, and David J. Foran. Multiple class segmentation using a unified framework over mean-shift patches. *CVPR*, 2007.

[288] C. Yanover and Y. Weiss. Finding the m most probable configurations in arbitrary graphical models. *Advances in Neural Information Processing Systems*, 16, 2004.

[289] Jinhui Yuan, Jianmin Li, and Bo Zhang. Exploiting spatial context constraints for automatic image region annotation. *Proc. ACM Multimedia*, pages 595–604, 2007.

[290] Jinhui Yuan, Jianmin Li, and Bo Zhang. Scene understanding with discriminative structured prediction. *CVPR*, 2008.

[291] Ramin Zabih and Vladimir Kolmogorov. Spatially coherent clustering using graph cuts. *CVPR*, 2004.

[292] C. Zahn. Graph theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computation*, 20:68–86, 1971.

[293] Y. J. Zhang. A survey on evaluation methods for image segmentation. *Pattern Recognition*, 29(8):1335–1346, 1996.

[294] Q. Zhi, G. Song, and J. Shi. Untangling cycles for contour groupings. *ICCV*, 2007.

[295] Long Zhu, Yuanhao Chen, Yuan Lin, Chenxi Lin, and Alan Yuille. Recursive segmentation and recognition templates for 2d parsing. *NIPS*, 2008.